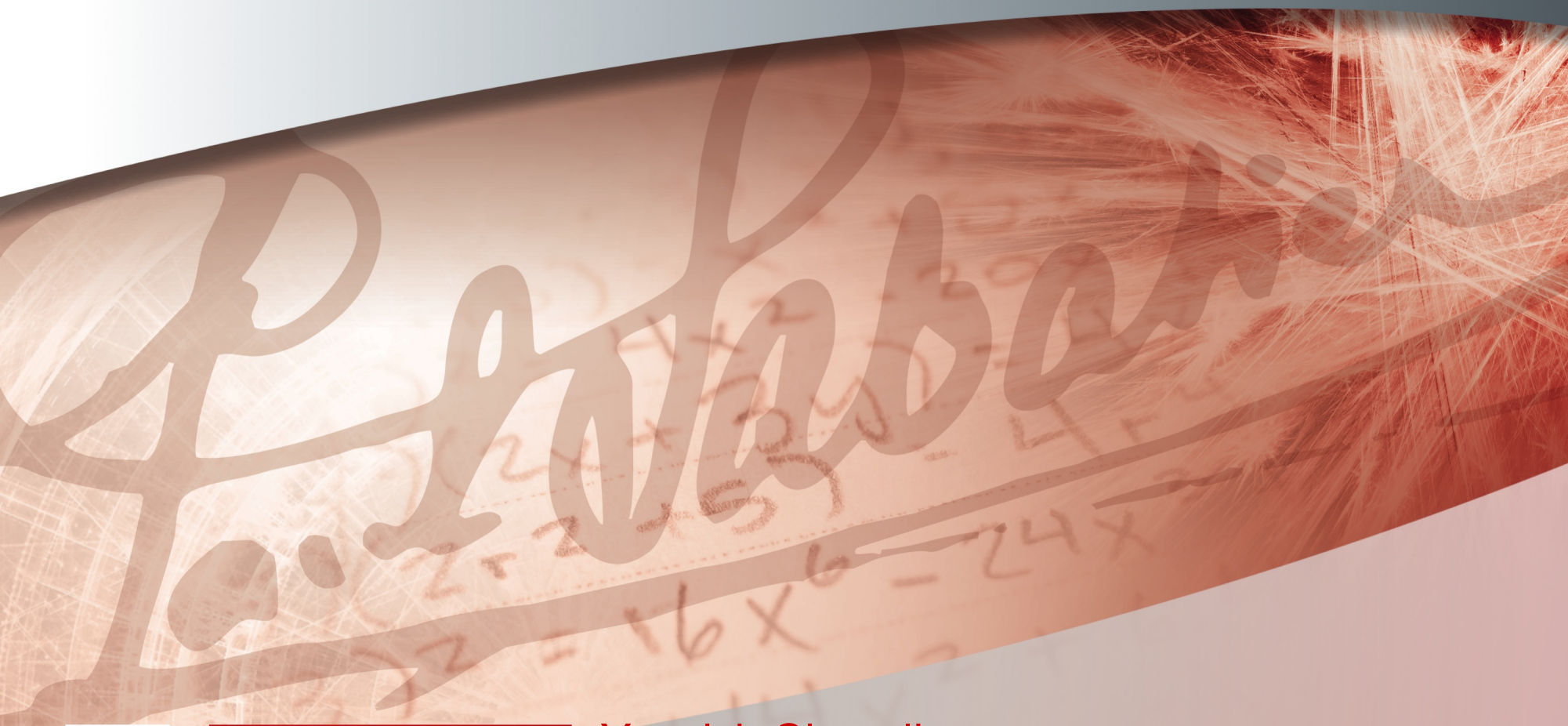
Sécurité des applications Web



Yannick Chevalier
Université de Toulouse
IUP NTIE M2 2012-2013

Plan

Domaine du cours

- Services Web
- Standards définis par W3C, Oasis Open, IETF, et soutenus par des vendeurs
- **Services Web :** services s'échangeant des documents XML
- **But :** interopérabilité sécurisée des services Web

Plan

Historique

Avant les années 1970

- Ordinateurs dans des environnements sécurisés
- Réinitialisation entre chaque programme

Débuts de la sécurité

- Apparition d'ordinateurs et de systèmes d'exploitations multi-tâches (~ 1970)
- But : garantir la même sécurité que dans la génération précédente

Historique

Avant les années 1970

- Ordinateurs dans des environnements sécurisés
- Réinitialisation entre chaque programme

Débuts de la sécurité

- Apparition d'ordinateurs et de systèmes d'exploitations multi-tâches (~ 1970)
- But : garantir la même sécurité que dans la génération précédente

SÉPARATION

SÉPARATION

L'exécution d'un programme ne doit pas influer sur l'exécution d'un autre programme

DÉFINITION EXTENSIVE

- Exemple : même si un programme boucle, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : même si un programme a un *memory leak*, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : dans un système multi-tâches, l'état du processeur lorsqu'il donne la main à un programme ne doit pas dépendre des autres programmes

SÉPARATION

SÉPARATION

L'exécution d'un programme ne doit pas influer sur l'exécution d'un autre programme

DÉFINITION EXTENSIVE

- Exemple : même si un programme boucle, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : même si un programme a un *memory leak*, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : dans un système multi-tâches, l'état du processeur lorsqu'il donne la main à un programme ne doit pas dépendre des autres programmes

SÉPARATION

SÉPARATION

L'exécution d'un programme ne doit pas influer sur l'exécution d'un autre programme

DÉFINITION EXTENSIVE

- Exemple : même si un programme boucle, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : même si un programme a un *memory leak*, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : dans un système multi-tâches, l'état du processeur lorsqu'il donne la main à un programme ne doit pas dépendre des autres programmes

Séparation

Séparation

L'exécution d'un programme ne doit pas influer sur l'exécution d'un autre programme

Définition extensive

- Exemple : même si un programme boucle, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : même si un programme a un *memory leak*, cela ne doit pas influer sur l'exécution des autres programmes
- Exemple : dans un système multi-tâches, l'état du processeur lorsqu'il donne la main à un programme ne doit pas dépendre des autres programmes

Base de confiance (1/2)

Assume/Guarantee

En informatique, on raisonne presque toujours par contrat entre différents composants :

- Chaque composant est écrit d'après des hypothèses sur ce que garantissent d'autres composants :
  - l'OS fait des hypothèses sur le matériel sous-jacent
  - les applications font des hypothèses sur l'OS
  - ...
- Et chaque composant garantit certaines propriétés :
  - absence de memory leaks,
  - temps d'exécution,
  - ...

Base de confiance (2/2)

Sécurité

- Formalismes exprimant les propriétés que les OS doivent satisfaire pour obtenir une sécurité équivalente à celle de la génération précédente
- Base de confiance : parties du système non-vérifiées qui permettent de garantir les propriétés qu'on recherche
- Exemple : effacement des registres par le processeur pour un système d'exploitation
- Note : pas forcément trivial lors de l'optimisation à la volée du code machine par le processeur

CONFIGURATION STATIQUE

PRINCIPE GÉNÉRAL

Les ressources sont allouées à chaque application de manière statique

APPLICATION À LA SÉCURITÉ

- Les politiques de contrôle d'accès sont définies avant l'exécution de chacun des composants
- Ces politiques ne peuvent pas changer tant que le système est en cours d'exécution

Contrôle d'accès statique

- En particulier, lors de l'exécution, il n'y a pas d'administrateur qui peut changer des droits
- On parle de politique de contrôle d'accès MAC :

Mandatory Access Control

- Permet de simplifier l'analyse de sécurité d'un système

CONFIGURATION STATIQUE

PRINCIPE GÉNÉRAL

Les ressources sont allouées à chaque application de manière statique

APPLICATION À LA SÉCURITÉ

- Les politiques de contrôle d'accès sont définies avant l'exécution de chacun des composants
- Ces politiques ne peuvent pas changer tant que le système est en cours d'exécution

Contrôle d'accès statique

- En particulier, lors de l'exécution, il n'y a pas d'administrateur qui peut changer des droits
- On parle de politique de contrôle d'accès MAC :

Mandatory Access Control

- Permet de simplifier l'analyse de sécurité d'un système

CONFIGURATION STATIQUE

PRINCIPE GÉNÉRAL

Les ressources sont allouées à chaque application de manière statique

APPLICATION À LA SÉCURITÉ

- Les politiques de contrôle d'accès sont définies avant l'exécution de chacun des composants
- Ces politiques ne peuvent pas changer tant que le système est en cours d'exécution

Contrôle d'accès statique

- En particulier, lors de l'exécution, il n'y a pas d'administrateur qui peut changer des droits
- On parle de politique de contrôle d'accès MAC :

Mandatory Access Control

- Permet de simplifier l'analyse de sécurité d'un système

CONFIGURATION STATIQUE

PRINCIPE GÉNÉRAL

Les ressources sont allouées à chaque application de manière statique

APPLICATION À LA SÉCURITÉ

- Les politiques de contrôle d'accès sont définies avant l'exécution de chacun des composants
- Ces politiques ne peuvent pas changer tant que le système est en cours d'exécution

Contrôle d'accès statique

- En particulier, lors de l'exécution, il n'y a pas d'administrateur qui peut changer des droits
- On parle de politique de contrôle d'accès MAC :

Mandatory Access Control

- Permet de simplifier l'analyse de sécurité d'un système

Configuration statique

Principe général

Les ressources sont allouées à chaque application de manière statique

Application à la sécurité

- Les politiques de contrôle d'accès sont définies avant l'exécution de chacun des composants
- Ces politiques ne peuvent pas changer tant que le système est en cours d'exécution

Contrôle d'accès statique

- En particulier, lors de l'exécution, il n'y a pas d'administrateur qui peut changer des droits
- On parle de politique de contrôle d'accès MAC :

Mandatory Access Control

- Permet de simplifier l'analyse de sécurité d'un système

SÉPARATION TEMPORELLE

ÉNONCÉ

La configuration doit spécifier le temps d'exécution alloué à chaque application

- Implémenté dans tous les systèmes temps réel dur
- Revient à allouer une fraction des cycles d'un processeur à chaque application
- Même lorsqu'une application plante, elle ne peut pas utiliser plus de cycles

Séparation spatiale

Énoncé

La configuration doit spécifier l'espace mémoire alloué à chaque application

- On parle aussi de partitionnement
- Implique l'absence de communication entre deux applications (IPC système V)
- En pratique :
  - On alloue une zone de la mémoire (RAM, DD,...) à chaque application
  - pas de malloc, ou un malloc sécurisé, allouant des zones mémoires au sein de la zone allouée
  - pas de pointeurs, sauf s'ils sont sécurisés pour ne lire que dans l'espace alloué
- Très difficile à mettre en œuvre car :
  - Besoin d'un compilateur ou d'un langage spécifique
  - Dans un système multi-tâches, les registres d'un processeurs sont forcément partagés entre plusieurs applications

Contrôle d'accès et séparation spatiale

- Difficile d'implémenter strictement des systèmes avec séparation
  - Communications entre composants
  - Importation de librairies partagées
- En pratique, violations du partitionnement spatial autorisé...
- ...mais ces violations sont encadrées par un mécanisme de contrôle
- On parle de systèmes de contrôle d'accès

Contrôle d'accès et séparation spatiale

- Difficile d'implémenter strictement des systèmes avec séparation
  - Communications entre composants
  - Importation de librairies partagées
- En pratique, violations du partitionnement spatial autorisé...
- ...mais ces violations sont encadrées par un mécanisme de contrôle
- On parle de systèmes de contrôle d'accès

Contrôle d'accès et séparation spatiale

- Difficile d'implémenter strictement des systèmes avec séparation
  - Communications entre composants
  - Importation de librairies partagées
- En pratique, violations du partitionnement spatial autorisé...
- ...mais ces violations sont encadrées par un mécanisme de contrôle
- On parle de systèmes de contrôle d'accès

Contrôle d'accès et séparation spatiale

- Difficile d'implémenter strictement des systèmes avec séparation
  - Communications entre composants
  - Importation de librairies partagées
- En pratique, violations du partitionnement spatial autorisé...
- ...mais ces violations sont encadrées par un mécanisme de contrôle
- On parle de systèmes de contrôle d'accès

Contrôle d'accès et séparation spatiale

- Difficile d'implémenter strictement des systèmes avec séparation
  - Communications entre composants
  - Importation de librairies partagées
- En pratique, violations du partitionnement spatial autorisé...
- ...mais ces violations sont encadrées par un mécanisme de contrôle
- On parle de systèmes de contrôle d'accès

Contrôle d'accès et séparation spatiale

- Difficile d'implémenter strictement des systèmes avec séparation
  - Communications entre composants
  - Importation de librairies partagées
- En pratique, violations du partitionnement spatial autorisé...
- ...mais ces violations sont encadrées par un mécanisme de contrôle
- On parle de systèmes de contrôle d'accès

Bons systèmes de contrôle d'accès

NEAT Security

- Non-bypassable : il est impossible de contourner le système de contrôle d'accès
- Evaluatable : il doit être possible d'évaluer (= décider) les conséquences d'une politique
- Always invoked : pas de "droits futurs", le système de contrôle doit être appelé lors de chaque accès
- Tamperproof : il est impossible d'altérer le fonctionnement du système de contrôle d'accès

- Ces critères de sécurité correspondent aussi à ce qu'on attend d'un système fiable (début cours sécurité)
- L'histoire du contrôle d'accès, c'est aussi les recherches pour échapper à ce modèle trop rigide
- Constante de la sécurité :
  - on sait sécuriser
  - mais il est difficile de concilier l'utilisabilité et la sécurité

Bons systèmes de contrôle d'accès

NEAT Security

- Non-bypassable : il est impossible de contourner le système de contrôle d'accès
- Evaluatable : il doit être possible d'évaluer (= décider) les conséquences d'une politique
- Always invoked : pas de "droits futurs", le système de contrôle doit être appelé lors de chaque accès
- Tamperproof : il est impossible d'altérer le fonctionnement du système de contrôle d'accès

- Ces critères de sécurité correspondent aussi à ce qu'on attend d'un système fiable (début cours sécurité)
- L'histoire du contrôle d'accès, c'est aussi les recherches pour échapper à ce modèle trop rigide
- Constante de la sécurité :
  - on sait sécuriser
  - mais il est difficile de concilier l'utilisabilité et la sécurité

Bons systèmes de contrôle d'accès

NEAT Security

- Non-bypassable : il est impossible de contourner le système de contrôle d'accès
- Evaluatable : il doit être possible d'évaluer (= décider) les conséquences d'une politique
- Always invoked : pas de "droits futurs", le système de contrôle doit être appelé lors de chaque accès
- Tamperproof : il est impossible d'altérer le fonctionnement du système de contrôle d'accès

- Ces critères de sécurité correspondent aussi à ce qu'on attend d'un système fiable (début cours sécurité)
- L'histoire du contrôle d'accès, c'est aussi les recherches pour échapper à ce modèle trop rigide
- Conséquence de la sécurité :

Bons systèmes de contrôle d'accès

NEAT Security

- Non-bypassable : il est impossible de contourner le système de contrôle d'accès
- Evaluatable : il doit être possible d'évaluer (= décider) les conséquences d'une politique
- Always invoked : pas de "droits futurs", le système de contrôle doit être appelé lors de chaque accès
- Tamperproof : il est impossible d'altérer le fonctionnement du système de contrôle d'accès

- Ces critères de sécurité correspondent aussi à ce qu'on attend d'un système fiable (début cours sécurité)
- L'histoire du contrôle d'accès, c'est aussi les recherches pour échapper à ce modèle trop rigide
- Constante de la sécurité :
  - on sait sécuriser
  - mais il est difficile de concilier l'utilisabilité et la sécurité

Bons systèmes de contrôle d'accès

NEAT Security

- Non-bypassable : il est impossible de contourner le système de contrôle d'accès
- Evaluatable : il doit être possible d'évaluer (= décider) les conséquences d'une politique
- Always invoked : pas de "droits futurs", le système de contrôle doit être appelé lors de chaque accès
- Tamperproof : il est impossible d'altérer le fonctionnement du système de contrôle d'accès

- Ces critères de sécurité correspondent aussi à ce qu'on attend d'un système fiable (début cours sécurité)
- L'histoire du contrôle d'accès, c'est aussi les recherches pour échapper à ce modèle trop rigide
- Constante de la sécurité :
  - on sait sécuriser
  - mais il est difficile de concilier l'utilisabilité et la sécurité

Bons systèmes de contrôle d'accès

NEAT Security

- Non-bypassable : il est impossible de contourner le système de contrôle d'accès
- Evaluatable : il doit être possible d'évaluer (= décider) les conséquences d'une politique
- Always invoked : pas de "droits futurs", le système de contrôle doit être appelé lors de chaque accès
- Tamperproof : il est impossible d'altérer le fonctionnement du système de contrôle d'accès

- Ces critères de sécurité correspondent aussi à ce qu'on attend d'un système fiable (début cours sécurité)
- L'histoire du contrôle d'accès, c'est aussi les recherches pour échapper à ce modèle trop rigide
- Constante de la sécurité :
  - on sait sécuriser
  - mais il est difficile de concilier l'utilisabilité et la sécurité

Plan

POURQUOI WEB ?

SERVICES UTILISANT LE PROTOCOLE HTTP

- Inconvénient : lourd, XML, couche supplémentaire
- Avantage : XML, permet de contourner les pare-feux, utilisables à partir d'un navigateur (browser)

SERVICES ET APPLICATIONS

- Pas réellement de différences pour un serveur
- Au pire, on peut qu'une application va en particulier servir des pages complètes alors qu'un service englobe tout ce qui répond à des requêtes HTTP

Dans ce cours, on verra comment sécuriser des applications accessibles à partir d'un browser.

Pourquoi Web ?

Services utilisant le protocole HTTP

- Inconvénient : lourd, XML, couche supplémentaire
- Avantage : XML, permet de contourner les pare-feux, utilisables à partir d'un navigateur (browser)

Services et applications

- Pas réellement de différences pour un serveur
- Au pire, on peut qu'une application va en particulier servir des pages complètes alors qu'un service englobe tout ce qui répond à des requêtes HTTP

Dans ce cours, on verra comment sécuriser des applications accessibles à partir d'un browser.

Pourquoi Web ?

Services utilisant le protocole HTTP

- Inconvénient : lourd, XML, couche supplémentaire
- Avantage : XML, permet de contourner les pare-feux, utilisables à partir d'un navigateur (browser)

Services et applications

- Pas réellement de différences pour un serveur
- Au pire, on peut qu'une application va en particulier servir des pages complètes alors qu'un service englobe tout ce qui répond à des requêtes HTTP

Dans ce cours, on verra comment sécuriser des applications accessibles à partir d'un browser.

Schéma général

3 parties :

1. Un client utilisant un browser pour émettre des *requêtes* vers un serveur
2. Un serveur recevant des requêtes et émettant des réponses
3. Un réseau transportant les requêtes et les réponses

Important

La sécurité du client n'est en général pas dans les mains du développeur d'un site Web, mais il est possible de parer certaines attaques dont il pourrait être victime.

Schéma général

3 parties :

1. Un client utilisant un browser pour émettre des *requêtes* vers un serveur
2. Un serveur recevant des requêtes et émettant des réponses
3. Un réseau transportant les requêtes et les réponses

Important

La sécurité du client n'est en général pas dans les mains du développeur d'un site Web, mais il est possible de parer certaines attaques dont il pourrait être victime.

Plan

OUTLINE

SÉCURITÉ DU CLIENT

- Architecture du client
- Partitionnement spatial
- Attaques liées au browser
- Quelques solutions possibles

Séparation spatiale

Principe d'un browser

- Il affiche un document à partir duquel il est possible de faire des requêtes vers des sites Webs
- Ce document contient du html, avec des requêtes formées à partir des balises `a`
- Il contient aussi des scripts, *i.e.*, des programmes qui peuvent générer des requêtes avec ou sans l'intervention de l'utilisateur

Sandboxing

Chaque fenêtre d'un browser est isolée de son environnement :

- Politique de *même origine* : un script sur une page servie par `example.com` ne peut effectuer des requêtes que vers le site `example.com`
- Exception : Websockets (plus tard)
- Partitionnement spatial : les browsers assurent que les scripts hébergés par différentes pages :
  1. ne peuvent pas avoir de variables communes
  2. ne partagent pas de code (mais copies de code autorisées)

Séparation spatiale

Principe d'un browser

- Il affiche un document à partir duquel il est possible de faire des requêtes vers des sites Webs
- Ce document contient du html, avec des requêtes formées à partir des balises `a`
- Il contient aussi des scripts, *i.e.*, des programmes qui peuvent générer des requêtes avec ou sans l'intervention de l'utilisateur

Sandboxing

Chaque fenêtre d'un browser est isolée de son environnement :

- Politique de *même origine* : un script sur une page servie par `example.com` ne peut effectuer des requêtes que vers le site `example.com`
- Exception : Websockets (plus tard)
- Partitionnement spatial : les browsers assurent que les scripts hébergés par différentes pages :
  1. ne peuvent pas avoir de variables communes
  2. ne partagent pas de codes de fonctions (mais copies de code autorisées)

Contenu d'une page Web

Base

- Une page Web est représentée par l'arbre des balises HTML qu'elle contient DOM
- Une balise spéciale, `script`, permet d'inclure en plus n'importe quel programme écrit en un langage reconnu par le browser
- Langage universellement reconnu : JavaScript (JS)

Modification d'une page Web

- Un script contenu dans une page peut modifier cette page
- Il peut ajouter, modifier, ou supprimer des balises de n'importe quel type
- Le browser réagit à ces modifications en temps réel

Contenu d'une page Web

Base

- Une page Web est représentée par l'arbre des balises HTML qu'elle contient DOM
- Une balise spéciale, `script`, permet d'inclure en plus n'importe quel programme écrit en un langage reconnu par le browser
- Langage universellement reconnu : JavaScript (JS)

Modification d'une page Web

- Un script contenu dans une page peut modifier cette page
- Il peut ajouter, modifier, ou supprimer des balises de n'importe quel type
- Le browser réagit à ces modifications en temps réel

Communications à partir d'une page Web (1/2)

Éléments HTML

- Certains éléments HTML effectuent naturellement des requêtes GET (`img`, `script`, `a`)
- Ces requêtes sont définies par une url qui est recherchée
- L'usage est d'encoder des éventuelles informations à passer à un serveur dans cette url
- Les scripts peuvent aussi envoyer des requêtes (*e.g.*, soumission d'un formulaire)

Communications à partir d'une page Web (2/2)

Requêtes à partir de JavaScript

- (universel) On crée un objet `XMLHttpRequest`
- On indique :
  - quelle méthode HTTP on désire utiliser (il n'y a pas que GET !)
  - l'url à contacter
  - éventuellement le contenu de la requête
  - la fonction à appeler lors de la réception de la réponse (inutile si la requête est synchrone)
- Et voilà !

OUTLINE

SÉCURITÉ DU CLIENT

- Architecture du client
- Partitionnement spatial
- Attaques liées au browser
- Quelques solutions possibles

Violations du partitionnement spatial (1/5)

Bibliothèques

- Il serait très gênant de demander à l'utilisateur de recharger une bibliothèque très répandue (*e.g.*, JQuery) chaque fois qu'il va sur un nouveau site
- Solution :
  - Utilisation du cache du browser
  - Utilisation d'un *Content Distribution Network* pour distribuer la version de la bibliothèque qu'on utilise
- Pour cela, on a besoin d'autoriser le chargement de scripts sur d'autres sites Web

Important

L'importation de scripts n'est jamais soumise à la politique de même origine

Violations du partitionnement spatial (1/5)

Bibliothèques

- Il serait très gênant de demander à l'utilisateur de recharger une bibliothèque très répandue (*e.g.*, JQuery) chaque fois qu'il va sur un nouveau site
- Solution :
  - Utilisation du cache du browser
  - Utilisation d'un *Content Distribution Network* pour distribuer la version de la bibliothèque qu'on utilise
- Pour cela, on a besoin d'autoriser le chargement de scripts sur d'autres sites Web

Important

L'importation de scripts n'est jamais soumise à la politique de même origine

Violations du partionnement spatial (2/5)

Extensions

- Les extensions des browsers peuvent définir des pages virtuelles
- Ces pages sont isolées des pages affichées (pour le code)
- Mais il est possible de faire des requêtes Javascript (XMLHttpRequest) entre ces pages et celles affichées

Cookies/localstorage

Stockage d'informations dans le browser

- Pas de différences pour la sécurité
- Définitions de domaine différente pour les cookies du reste
- Le code servi par un domaine ne peut utiliser que des informations définies par ce domaine

Violations du partionnement spatial (3/5)

La sécurité, c'est embêtant !

D'après ce qui a été écrit, il n'est pas possible

- de faire référence à une image (une ressource) hébergée par un autre domaine
- d'envoyer des informations personnelles sur l'utilisateur à un autre site (*e.g.*, `doubleclick.com`)

Cross-Origin Resource Sharing (CORS)

- Les scripts situés sur la page servie par le domaine A peuvent effectuer des requêtes vers le domaine B
- Mais pour avoir une réponse, le domaine B doit avoir été configuré pour accepter explicitement les requêtes venant du domaine A (ou de tout domaine)

Important

Politique de contrôle d'accès à compléter éventuellement au sein domaine B avec les informations propres de l'utilisateur

Violations du partionnement spatial (3/5)

La sécurité, c'est embêtant !

D'après ce qui a été écrit, il n'est pas possible

- de faire référence à une image (une ressource) hébergée par un autre domaine
- d'envoyer des informations personnelles sur l'utilisateur à un autre site (*e.g.*, `doubleclick.com`)

Cross-Origin Resource Sharing (CORS)

- Les scripts situés sur la page servie par le domaine A peuvent effectuer des requêtes vers le domaine B
- Mais pour avoir une réponse, le domaine B doit avoir été configuré pour accepter explicitement les requêtes venant du domaine A (ou de tout domaine)

Important

Politique de contrôle d'accès à compléter éventuellement au sein domaine B avec les informations liées de l'utilisateur

Violations du partionnement spatial (3/5)

La sécurité, c'est embêtant !

D'après ce qui a été écrit, il n'est pas possible

- de faire référence à une image (une ressource) hébergée par un autre domaine
- d'envoyer des informations personnelles sur l'utilisateur à un autre site (*e.g.*, `doubleclick.com`)

Cross-Origin Resource Sharing (CORS)

- Les scripts situés sur la page servie par le domaine A peuvent effectuer des requêtes vers le domaine B
- Mais pour avoir une réponse, le domaine B doit avoir été configuré pour accepter explicitement les requêtes venant du domaine A (ou de tout domaine)

Important

Politique de contrôle d'accès à compléter éventuellement au sein domaine B avec les informations personnelles de l'utilisateur

VIOLATIONS DU PARTIONNEMENT SPATIAL (4/5)

RAPPEL

- Les scripts peuvent être servis par n'importe quel site
- Donc on peut effectuer une requête vers n'importe quel site en utilisant une balise `script` avec une url forgée correctement

JSON/JSONP

- JSON : codage d'objets JavaScript simples en du texte
- JSONP : enrobage du texte pour fournir un code JS valide (appel d'une fonction)
- Utilisation :
  - insertion d'une balise `script` dans la page avec une adresse codant les informations qu'on désire envoyer
  - on laisse le navigateur *évaluer* le code JS reçu en réponse de la requête

VIOLATIONS DU PARTIONNEMENT SPATIAL (4/5)

RAPPEL

- Les scripts peuvent être servis par n'importe quel site
- Donc on peut effectuer une requête vers n'importe quel site en utilisant une balise `script` avec une url forgée correctement

JSON/JSONP

- JSON : codage d'objets JavaScript simples en du texte
- JSONP : enrobage du texte pour fournir un code JS valide (appel d'une fonction)
- Utilisation :
  - insertion d'une balise `script` dans la page avec une adresse codant les informations qu'on désire envoyer
  - on laisse le navigateur *évaluer* le code JS reçu en réponse de la requête

VIOLATION DU PARTITIONNEMENT SPATIAL (5/5)

FRAME/IFRAME

- Élément html permettant d'inclure un autre site
- Les DOM sont séparés correctement

VIOLATION DE PARTITION

- Gestion complexe de l'affichage (à quels éléments un pixel est relié)
- Transparence
- En pratique, une action de l'utilisateur peut toucher plusieurs éléments, y compris en dehors et à l'intérieur d'une (i)frame

VIOLATION DU PARTITIONNEMENT SPATIAL (5/5)

FRAME/IFRAME

- Élément html permettant d'inclure un autre site
- Les DOM sont séparés correctement

VIOLATION DE PARTITION

- Gestion complexe de l'affichage (à quels éléments un pixel est relié)
- Transparence
- En pratique, une action de l'utilisateur peut toucher plusieurs éléments, y compris en dehors et à l'intérieur d'une (i)frame

OUTLINE

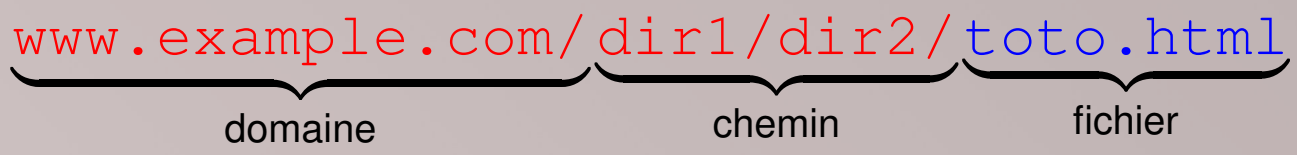
SÉCURITÉ DU CLIENT

- Architecture du client
- Partitionnement spatial
- Attaques liées au browser
- Quelques solutions possibles

COOKIES

DÉFINITION DU DOMAINE

- Par défaut, un cookie est lié à un domaine et un chemin dans ce domaine :



- Par défaut, le protocole utilisé n'est pas pris en compte
- Secure Cookie : en plus, il faut utiliser le protocole HTTP
- Il est possible de spécifier un autre domaine pour le cookie (third-party cookies), mais bloquables par l'utilisateur

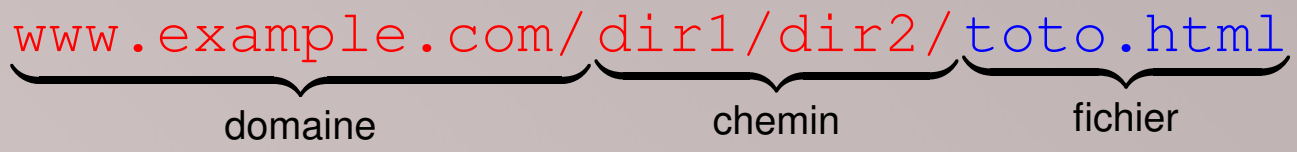
IMPORTANT

La politique de gestion des cookies est établie sur des bases différentes de celle des autres ressources, ce qui est une source potentielle de conflit

COOKIES

DÉFINITION DU DOMAINE

- Par défaut, un cookie est lié à un domaine et un chemin dans ce domaine :



- Par défaut, le protocole utilisé n'est pas pris en compte
- Secure Cookie : en plus, il faut utiliser le protocole HTTP
- Il est possible de spécifier un autre domaine pour le cookie (third-party cookies), mais bloquables par l'utilisateur

IMPORTANT

La politique de gestion des cookies est établie sur des bases différentes de celle des autres ressources, ce qui est une source potentielle de conflit

Cross-Site Request Forgery

Non-blocage des images + cookies

- À la différence des localStorage, pour lesquels cela demande un peu plus de travail, les cookies d'un domaine sont envoyés lors de chaque requête vers ce domaine.
- La page d'un domaine A peut donc faire une requête vers un domaine B en utilisant les cookies (*e.g.* identification de l'utilisateur) de ce domaine
- Si le domaine B ne vérifie pas que la requête est issue d'une page qu'il a servie, et si cette requête est valide, il va l'accepter.

CSRF

- Un site malicieux utilise le fait que l'utilisateur est potentiellement logué sur un autre site
- Il envoie des requêtes à cet autre site qui seront accompagnées des cookies de cet autre site (donc authentifiées)
- Ex : faire un virement si l'autre site est celui d'une banque, récupérer une liste de contact, etc.

Cross-Site Request Forgery

Non-blocage des images + cookies

- À la différence des localStorage, pour lesquels cela demande un peu plus de travail, les cookies d'un domaine sont envoyés lors de chaque requête vers ce domaine.
- La page d'un domaine A peut donc faire une requête vers un domaine B en utilisant les cookies (*e.g.* identification de l'utilisateur) de ce domaine
- Si le domaine B ne vérifie pas que la requête est issue d'une page qu'il a servie, et si cette requête est valide, il va l'accepter.

CSRF

- Un site malicieux utilise le fait que l'utilisateur est potentiellement logué sur un autre site
- Il envoie des requêtes à cet autre site qui seront accompagnées des cookies de cet autre site (donc authentifiées)
- Ex : faire un virement si l'autre site est celui d'une banque, récupérer une liste de contact, etc.

Login CSRF

Attaque dépendant du browser utilisé

Partitionnement spatial lors de la visualisation

- un site malicieux peut superposer une image opaque pour couvrir une iframe
- L'image contient un captcha
- L'iframe contient la page de login d'un site (Yahoo !, Google, etc.)

Login CSRF

- Le captcha demande à l'utilisateur de rentrer le login et le mot de passe de l'attaquant
- Une fois l'utilisateur loggué, l'attaquant récupère sur son compte les données de navigation de l'utilisateur

Login CSRF

Attaque dépendant du browser utilisé

Partitionnement spatial lors de la visualisation

- un site malicieux peut superposer une image opaque pour couvrir une iframe
- L'image contient un captcha
- L'iframe contient la page de login d'un site (Yahoo !, Google, etc.)

Login CSRF

- Le captcha demande à l'utilisateur de rentrer le login et le mot de passe de l'attaquant
- Une fois l'utilisateur loggué, l'attaquant récupère sur son compte les données de navigation de l'utilisateur

Interception de données

Problème de partitionnement spatial

- Couverture d'une iframe par un élément presque transparent
- Les événements utilisateurs sont aussi envoyés à cet élément
- Permet au site hôte de récupérer les données envoyées au site hébergé dans l'iframe

Peu de solutions

- Beaucoup de sites utilisent de manière critique le recouvrement par des éléments transparents
- Assurer le partitionnement spatial des pixels serait très gênant (popup html, listes descendantes)
- Pas de solutions programmées

Interception de données

Problème de partitionnement spatial

- Couverture d'une iframe par un élément presque transparent
- Les événements utilisateurs sont aussi envoyés à cet élément
- Permet au site hôte de récupérer les données envoyées au site hébergé dans l'iframe

Peu de solutions

- Beaucoup de sites utilisent de manière critique le recouvrement par des éléments transparents
- Assurer le partitionnement spatial des pixels serait très gênant (popup html, listes descendantes)
- Pas de solutions programmées

JSONP

- L'utilisation de JSONP nécessite une bonne collaboration entre les 2 domaines
- Mais à cause de l'évaluation, le domaine B obtient tous les pouvoirs (parce qu'il peut renvoyer ce qu'il veut à la requête, ce sera évalué) sur la page affichée et ses données
- Autrement dit, il faut avoir une confiance absolue dans le domaine B pour lui faire des requêtes *via* JSONP

OUTLINE

SÉCURITÉ DU CLIENT

- Architecture du client
- Partitionnement spatial
- Attaques liées au browser
- Quelques solutions possibles

IFRAMES

- Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE

```
Header always append X-Frame-Options SAMEORIGIN
Header always append X-Frame-Options DENY
```

- Dans le premier cas, les iframes sont autorisées sur le même site
- Dans le second, elles sont toujours refusées
- Header des réponses à un GET, honoré par tous les browsers récents

IFRAMES

- Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE

```
Header always append X-Frame-Options SAMEORIGIN
Header always append X-Frame-Options DENY
```

- Dans le premier cas, les iframes sont autorisées sur le même site
- Dans le second, elles sont toujours refusées
- Header ... : réponses à un GET, honoré par [illegible]

IFRAMES

- Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE

```
Header always append X-Frame-Options SAMEORIGIN
Header always append X-Frame-Options DENY
```

- Dans le premier cas, les iframes sont autorisées sur le même site
- Dans le second, elles sont toujours refusées
- Header des réponses à un GET, honoré par tous les browsers récents

IFRAMES

- Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE

```
Header always append X-Frame-Options SAMEORIGIN
```

```
Header always append X-Frame-Options DENY
```

- Dans le premier cas, les iframes sont autorisées sur le même site
- Dans le second, elles sont toujours refusées
- Header des réponses à un GET, honoré par tous les browsers récents

IFRAMES

- Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE

```
Header always append X-Frame-Options SAMEORIGIN
Header always append X-Frame-Options DENY
```

- Dans le premier cas, les iframes sont autorisées sur le même site
- Dans le second, elles sont toujours refusées
- Header des réponses à un GET, honoré par tous les browsers récents

CSRF

Base des CSRF

- L'attaquant devine que l'utilisateur est loggué sur un certain site

  on n'y peut rien s'il devine correctement

- L'attaquant peut construire des requêtes ayant du sens

  Par contre, on peut s'arranger pour que les requêtes incluent un élément aléatoire non-devinable

Empêchement des requêtes CSRF

- Demander au serveur de regarder la page d'origine de la requête

  utile, mais pas suffisant surtout si le site peut être dans une iframe

- Ajouter un cookie de session aléatoire lorsqu'un utilisateur est loggué

CSRF

Base des CSRF

- L'attaquant devine que l'utilisateur est loggué sur un certain site

  on n'y peut rien s'il devine correctement

- L'attaquant peut construire des requêtes ayant du sens

  Par contre, on peut s'arranger pour que les requêtes incluent un élément aléatoire non-devinable

Empêchement des requêtes CSRF

- Demander au serveur de regarder la page d'origine de la requête

  utile, mais pas suffisant surtout si le site peut être dans une iframe

- Ajouter un cookie de session aléatoire lorsqu'un utilisateur est loggué

Empêchement des requêtes CSRF

Spécifique à chaque serveur d'application

Important

Application sécurisée par défaut

Exemple (Rails), app/controllers/application_controller.rb

```
protect_from_forgery
```

Liste d'attaques

http://guides.rubyonrails.org/security.html

Empêchement des requêtes CSRF

Spécifique à chaque serveur d'application

Important

Application sécurisée par défaut

Exemple (Rails), app/controllers/application_controller.rb

```
protect_from_forgery
```

Liste d'attaques

http://guides.rubyonrails.org/security.html

Empêchement des requêtes CSRF

Spécifique à chaque serveur d'application

Important

Application sécurisée par défaut

Exemple (Rails), app/controllers/application_controller.rb

```
protect_from_forgery
```

Liste d'attaques

http://guides.rubyonrails.org/security.html

INCLUSION DE SCRIPTS TIERS

LA BATAILLE PERDUE ?

IMPORTANT

Un script importé peut contrôler entièrement la page Web.

EXTENSIONS

- Ajouts à un browser
- Les extensions de type *content script* sont intégrées aux pages Web
- Un utilisateur malicieux peut donc monter des attaques en :
  - Créant une extension spécifique et en l'intégrant à son browser
  - Puis en visitant le site visé

CONCLUSION

- Le serveur ne peut pas faire confiance en les requêtes qu'il reçoit
- Mais le serveur peut s'arranger pour que le client ait confiance en lui

Inclusion de scripts tiers

La bataille perdue ?

Important

Un script importé peut contrôler entièrement la page Web.

Extensions

- Ajouts à un browser
- Les extensions de type *content script* sont intégrées aux pages Web
- Un utilisateur malicieux peut donc monter des attaques en :
  - Créant une extension spécifique et en l'intégrant à son browser
  - Puis en visitant le site visé

Conclusion

- Le serveur ne peut pas faire confiance en les requêtes qu'il reçoit
- Mais le serveur peut s'arranger pour que le client ait confiance en lui

Inclusion de scripts tiers

La bataille perdue ?

Important

Un script importé peut contrôler entièrement la page Web.

Extensions

- Ajouts à un browser
- Les extensions de type *content script* sont intégrées aux pages Web
- Un utilisateur malicieux peut donc monter des attaques en :
  - Créant une extension spécifique et en l'intégrant à son browser
  - Puis en visitant le site visé

Conclusion

- Le serveur ne peut pas faire confiance en les requêtes qu'il reçoit
- Mais le serveur peut s'arranger pour que le client ait confiance en lui

Inclusion de scripts tiers

La bataille perdue ?

Important

Un script importé peut contrôler entièrement la page Web.

Extensions

- Ajouts à un browser
- Les extensions de type *content script* sont intégrées aux pages Web
- Un utilisateur malicieux peut donc monter des attaques en :
  - Créant une extension spécifique et en l'intégrant à son browser
  - Puis en visitant le site visé

Conclusion

- Le serveur ne peut pas faire confiance en les requêtes qu'il reçoit
- Mais le serveur peut s'arranger pour que le client ait confiance en lui

Plan

Outline

Sécurité du réseau

- Principaux protocoles utilisés
- TLS
- Sécurisation des messages SOAP
- Encryption et chiffrement

Communication entre le browser et un site

HTTP

- Communications non-sécurisées
- Il est relativement facile pour un attaquant de lire des échanges et d'envoyer de fausses requêtes

HTTPS

- Requête HTTP GET pour un upgrade avec utilisation de TLS
- Changement de port (443 au lieu de 80)
- Négociation de clef et authentification
- Suite des communications cryptées

Important

Les requêtes JS passent par HTTP/HTTPS

Communication entre le browser et un site

HTTP

- Communications non-sécurisées
- Il est relativement facile pour un attaquant de lire des échanges et d'envoyer de fausses requêtes

HTTPS

- Requête HTTP GET pour un upgrade avec utilisation de TLS
- Changement de port (443 au lieu de 80)
- Négociation de clef et authentification
- Suite des communications cryptées

Important

Les requêtes JS passent par HTTP/HTTPS

WEBSOCKET

TRAITS PRINCIPAUX

- Protocole indépendant de HTTP
- Connection sur le port 80
- Moins d'encodage, donc plus rapide qu'HTTP
- Passage à ce protocole avec un GET + upgrade
- Utilisation avec TCP (`ws://`) ou TLS (`wss://`

NÉGOCIATION

Dans une phase de Rendez-vous (handshake) le client et le serveur :

- Présentent des certificats assurant de leur identité
- Vérifient que ces certificats appartiennent bien à leur pair (*i.e.*, que la clef secrète correspondant au certificat est bien connue)
- Vérifient que ces certificats sont délivrés par des autorités de confiance
- Négocient un algorithme de chiffrement et une clef partagée

IMPORTANT

- Cette phase de négociation peut être renouvelée périodiquement
- Après négociation, la clef chiffre tous les messages échangés entre le client et le serveur

NÉGOCIATION

Dans une phase de Rendez-vous (handshake) le client et le serveur :

- Présentent des certificats assurant de leur identité
- Vérifient que ces certificats appartiennent bien à leur pair (*i.e.*, que la clef secrète correspondant au certificat est bien connue)
- Vérifient que ces certificats sont délivrés par des autorités de confiance
- Négocient un algorithme de chiffrement et une clef partagée

IMPORTANT

- **Cette phase de négociation peut être renouvelée périodiquement**
- **Après négociation, la clef chiffre tous les messages échangés entre le client et le serveur**

Outline

Sécurité du réseau

- Principaux protocoles utilisés
- TLS
- Sécurisation des messages SOAP
- Encryption et chiffrement

Attaques cryptographiques sur TLS/SSL

Important

Il existe des attaques sur tous les modes d'utilisation de TLS 1.0

Points gênants

- Beaucoup d'attaques découvertes depuis fin 2011
- Dans la plupart des cas, il s'agit d'attaques connues depuis longtemps sur TLS, mais qui avaient été contrées

  Le contenu chiffré ou les clients et serveurs étaient adaptés
- Le problème vient de l'utilisation de protocoles spécifiques à l'intérieur de TLS (HTTP, Websocket)

Attaques cryptographiques sur TLS/SSL

Important

Il existe des attaques sur tous les modes d'utilisation de TLS 1.0

Points gênants

- Beaucoup d'attaques découvertes depuis fin 2011
- Dans la plupart des cas, il s'agit d'attaques connues depuis longtemps sur TLS, mais qui avaient été contrées

  Le contenu chiffré ou les clients et serveurs étaient adaptés
- Le problème vient de l'utilisation de protocoles spécifiques à l'intérieur de TLS (HTTP, Websocket)

CONTRE-MESURES

BROWSERS

- Probable que les browsers et les serveurs se mettent à supporter TLS 1.1 et 1.2
- Changement des bibliothèques pour en utiliser des versions sécurisées

Interlude

Échange réel entre développeurs

Un développeur Facebook (wesbos) demande :

Hey guys, I'm using the freshly downloaded PHP API
I've got eveything setup and when I login to authenticate, I get this error :
Fatal error : Uncaught CurlException : 60 : SSL certificate problem, verify that the CA cert is OK. Details : error :14090086 :SSL routines :SSL3_GET_SERVER_CERTIFICATE :certificate verify failed thrown in /Users/wesbos/Dropbox/OrangeRhinoMedia/reporting/src/facebook.php on line 512

Interlude (suite)

Réponse de Telemako :

[...] try this one :

```
$opts[CURLOPT_SSL_VERIFYPEER] = false ;
$opts[CURLOPT_SSL_VERIFYHOST] = 2 ;
```

Commentaire de Phileo :

Is there any good reason why these shouldn't be a part of the core default $CURL_OPTS ?

Sur un autre forum

When you connect to a remote server with SSL, their cert might be invalid, expired, or not signed by a recognized CA. I've never used the "verifyhost" option but *I often have to set verifypeer = 0 (false)* when the remote site is too cheap to buy a real cert for their site.

Interlude (suite)

Réponse de Telemako :

[...] try this one :

$opts[CURLOPT_SSL_VERIFYPEER] = false ;

$opts[CURLOPT_SSL_VERIFYHOST] = 2 ;

Commentaire de Phileo :

Is there any good reason why these shouldn't be a part of the core default $CURL_OPTS ?

Sur un autre forum

When you connect to a remote server with SSL, their cert might be invalid, expired, or not signed by a recognized CA. I've never used the "verifyhost" option but *I often have to set verifypeer = 0 (false)* when the remote site is too cheap to buy a real cert for their site.

Interlude (suite)

Réponse de Telemako :

[...] try this one :

$opts[CURLOPT_SSL_VERIFYPEER] = false ;

$opts[CURLOPT_SSL_VERIFYHOST] = 2 ;

Commentaire de Phileo :

Is there any good reason why these shouldn't be a part of the core default $CURL_OPTS ?

Sur un autre forum

When you connect to a remote server with SSL, their cert might be invalid, expired, or not signed by a recognized CA. I've never used the "verifyhost" option but *I often have to set verifypeer = 0 (false)* when the remote site is too cheap to buy a real cert for their site.

COMMENTAIRES

VERIFY ?

- Verify host sert à vérifier que le certificat que le serveur présente au client correspond bien au nom du serveur
- Verify Peer sert à vérifier que ce certificat est enregistré, et non auto-signé

UTILITÉ

- Demander à vérifier l'hôte et son certificat sont la seule protection contre une attaque MITM
- On va voir comment configurer correctement un serveur
- Mais on ne peut rien faire contre la bêtise d'un développeur

COMMENTAIRES

VERIFY ?

- Verify host sert à vérifier que le certificat que le serveur présente au client correspond bien au nom du serveur
- Verify Peer sert à vérifier que ce certificat est enregistré, et non auto-signé

UTILITÉ

- Demander à vérifier l'hôte et son certificat sont la seule protection contre une attaque MITM
- On va voir comment configurer correctement un serveur
- Mais on ne peut rien faire contre la bêtise d'un développeur

Danger de TLS

- Beaucoup d'autres services dépendent de la sécurité de TLS
- Mea Culpa :

COURS DES ANNÉES PRÉCÉDENTES

Utiliser TLS c'est résoudre 99% des problèmes

- Vu les erreurs des développeurs (pour les mash up) et les failles cryptographiques, on ne peut plus être aussi affirmatif
- Complaisance des *industriels* : les chiffrements/signatures des messages sont plus difficile à traiter que leur insertion dans un canal sécurisé comme TLS (*cf.* OAuth *vs.* OAuth 2)

Danger de TLS

- Beaucoup d'autres services dépendent de la sécurité de TLS
- Mea Culpa :

COURS DES ANNÉES PRÉCÉDENTES

Utiliser TLS c'est résoudre 99% des problèmes

- Vu les erreurs des développeurs (pour les mash up) et les failles cryptographiques, on ne peut plus être aussi affirmatif
- Complaisance des *industriels* : les chiffrements/signatures des messages sont plus difficile à traiter que leur insertion dans un canal sécurisé comme TLS (*cf.* OAuth *vs.* OAuth 2)

Danger de TLS

- Beaucoup d'autres services dépendent de la sécurité de TLS
- Mea Culpa :

Cours des années précédentes

Utiliser TLS c'est résoudre 99% des problèmes

- Vu les erreurs des développeurs (pour les mash up) et les failles cryptographiques, on ne peut plus être aussi affirmatif
- Complaisance des *industriels* : les chiffrements/signatures des messages sont plus difficile à traiter que leur insertion dans un canal sécurisé comme TLS (*cf.* OAuth *vs.* OAuth 2)

Danger de TLS

- Beaucoup d'autres services dépendent de la sécurité de TLS
- Mea Culpa :

Cours des années précédentes

Utiliser TLS c'est résoudre 99% des problèmes

- Vu les erreurs des développeurs (pour les mash up) et les failles cryptographiques, on ne peut plus être aussi affirmatif
- Complaisance des *industriels* : les chiffrements/signatures des messages sont plus difficile à traiter que leur insertion dans un canal sécurisé comme TLS (*cf.* OAuth *vs.* OAuth 2)

Danger de TLS

- Beaucoup d'autres services dépendent de la sécurité de TLS
- Mea Culpa :

Cours des années précédentes

Utiliser TLS c'est résoudre 99% des problèmes

- Vu les erreurs des développeurs (pour les mash up) et les failles cryptographiques, on ne peut plus être aussi affirmatif
- Complaisance des *industriels* : les chiffrements/signatures des messages sont plus difficile à traiter que leur insertion dans un canal sécurisé comme TLS (*cf.* OAuth *vs.* OAuth 2)

CONFIGURATION DU SERVICE

SERVICE.XML

```
<transports>
 <transport>https</transport>
</transports>
```

- Le WSDL sera généré correctement à partir de ce fichier
- On peut avoir plusieurs éléments transport

TLS avec Axis2

Configuration globale

axis2.xml (à décommenter)

```
<!-transportSender name="https" class="axis2_http_sender">
 <parameter name="PROTOCOL" locked="false">HTTP/1.1</parameter>
 <parameter name="xml-declaration" insert="false"/>
</transportSender>
<parameter
name="SERVER_CERT">/path/to/ca/certificate</parameter> (obligatoire)
<parameter
name="KEY_FILE">/path/to/client/certificate/chain/file</parameter
(optionnel)
<parameter name="SSL_PASSPHRASE">passphrase</parameter> (optionnel)
->
```

TLS avec Axis2

Notes sur la configuration globale

- sous Ubuntu, `/usr/lib/axis2/axis2.conf`
- configuration de Apache (httpd.conf) pas décrite
- SERVER_CERT contient un certificat du serveur certifié par un CA reconnu par le client, à configurer avec Apache

  Il faut un certificat par nom de domaine !
- KEY_FILE contient un couple clef publique/clef privée pour le client
- SSL_PASSPHRASE sert à chiffrer le fichier qui contient ce couple de clef

LIMITATIONS APPLICATIVES DE TLS

SÉCURITÉ POINT-À-POINT

- Garanti les communications entre 2 endpoints
- Ne garanti toujours pas les identités des endpoints
- Choix à faire : TLS avec ou sans authentification
- Pas de sécurité bout-à-bout : relais (SOAP) malhonnêtes

Limitations de TLS

Limitations applicatives

Problèmes applicatifs

- TLS = cryptage inter-messages d'un flux de données
- Pas vraiment possible de conserver des logs sécurisés
- C'est un problème pour tout se qui ressemble à un contrat (repudiation)
- authentification d'un client optionnelle, et nécessite d'avoir enregistré ses clefs publiques/privées

Problème d'architecture

- Séparation dans le serveur entre la partie applicative (qui gère les fournisseurs de services) et la partie réseau
- Re-négociations au sein de TLS : il n'est pas possible de garantir à une application qu'un agent authentifié par TLS lors d'un premier message communique ensuite avec la même authentification (similaire au cross-site scripting)

Pour remédier à ces limitations, besoin de cryptographie au sein des messages

OUTLINE

SÉCURITÉ DU RÉSEAU

- Principaux protocoles utilisés
- TLS
- Sécurisation des messages SOAP
- Encryption et chiffrement

PRINCIPE

- On veut *annoter* des spécifications de services par des contraintes de sécurité
- WS-Policy définit un cadre général pour l'écriture de contraintes
- Essentiellement,
  - des nœuds `all`, dont tous les fils doivent être satisfait
  - des nœuds `any`, dont au moins un des les fils doit être satisfait
- Une politique peut être attachée à n'importe quel nœud d'un fichier WSDL (WS-I, pas standard)
- L'attachement se fait
  - soit en mettant un nœud `Policy` (mal vu)
  - soit une référence `PolicyReference` (soit vers l'extérieur, soit vers un nœud `Policy` fils de `definitions`)

WS-ReliableMessaging

But :

garantir l'ordre des messages entre l'émetteur et le récepteur

Configuration : service.xml

```
<module ref="sandesha2" />
<module ref="addressing" />
```

- module Sandesha2
- méthode : compteur + ack
- addressing est utilisé pour définir des sessions avec un endpoint (garantie d'ordre inutile s'il n'y a pas de session !)
- Permet d'éviter certaines attaques (faibles)
- Surtout, permet d'avoir une meilleure qualité de service

WS-ReliableMessaging

But :

garantir l'ordre des messages entre l'émetteur et le récepteur

Configuration : service.xml

```
 <module ref="sandesha2" />
<module ref="addressing" />
```

- module Sandesha2
- méthode : compteur + ack
- addressing est utilisé pour définir des sessions avec un endpoint (garantie d'ordre inutile s'il n'y a pas de session !)
- Permet d'éviter certaines attaques (faibles)
- Surtout, permet d'avoir une meilleure qualité de service

WS-SecureConversation

Buts :

- Établissement d'un *contexte de sécurité* pour des séquences de message
- Comprend des identificateurs de sessions, mais aussi l'établissement de clefs partagées

- Dans le fichier de politique de sécurité, préciser qu'on veut utiliser des clefs dérivées, implicitement ou explicitement
- Rampart comprend alors qu'il doit utiliser WS-SC pour obtenir ces clefs
- Négociations possibles : bootstrap, amend, renew, cancel
- Pour chacune de ces négociations on peut spécifier une politique

WS-SecureConversation

Buts :

- Établissement d'un *contexte de sécurité* pour des séquences de message
- Comprend des identificateurs de sessions, mais aussi l'établissement de clefs partagées

- Dans le fichier de politique de sécurité, préciser qu'on veut utiliser des clefs dérivées, implicitement ou explicitement
- Rampart comprend alors qu'il doit utiliser WS-SC pour obtenir ces clefs
- Négociations possibles : bootstrap, amend, renew, cancel
- Pour chacune de ces négociations on peut spécifier une politique

WS-SecurityPolicy

But :

spécifier les contraintes de sécurité sur les messages

- Constructions de base :
  - des Tokens, qui véhiculent des assertions
  - des clefs, qui protègent ces assertions et des parties du message
- Les tokens sont placés dans le header du message
- Ils sont de différents types
- Les clefs doivent être nommées, ou leur construction doit être décrite (clefs dérivées)

WS-SecurityPolicy

Types de Token

- Username token : identification d'un utilisateur (avec mot de passe, etc.)
- Certificats X.509 (clefs publiques)
- Kerberos : implique l'utilisation du protocole Kerberos (par le client ou le serveur) pour récupérer un token satisfaisant la politique et l'inclure dans le message
- SecureConversationToken : implique l'utilisation de WS-SecureConversation. Il faut alors une sous-politique pour décrire les conditions de négociation de ce token
- SAMLToken : certificat SAML
- HTTPSToken : authentification serveur et/ou client, implique l'utilisation de https

Sauf pour des cas très simple, nécessite la mise en place d'un *Security Token Server*. Code à développer à la main...

Security Token Service

Rapidement :

- Il faut implémenter à la main des interfaces
- ces classes seront ensuite utilisées par Rampart lors des vérifications de message
- Typiquement,
  - besoin de se connecter à une base de données utilisateur/mot de passe
  - besoin de se connecter à une base de clefs
- Les configurations classiques sont automatisables (service par défaut avec juste les BD à remplir)

Documentation

- WSO2 Web Service Application Framework
- Basé sur Axis2 (pour la base)

Note sur TLS

Services REST ?

Du point de vue de développeur, les services REST sont :

- Plus faciles à comprendre
- Plus rapides à mettre en œuvre
- Les plus utilisés pour créer des applications (RoR, node.js, wordpress,. . .)

Côté obscur des services REST

- Pas de standardisation
- Les clients et serveurs sont écrits par le développeur d'un site Web à coups d'XMLHttpRequest
- Le développeur moyen est incapable d'ajouter une interface cryptographique propre, et encore moins un protocole correct
- Quelques librairies existent (pour OAuth v1), mais elles sont rares

Note sur TLS

Services REST ?

Du point de vue de développeur, les services REST sont :

- Plus faciles à comprendre
- Plus rapides à mettre en œuvre
- Les plus utilisés pour créer des applications (RoR, node.js, wordpress,. . .)

Côté obscur des services REST

- Pas de standardisation
- Les clients et serveurs sont écrits par le développeur d'un site Web à coups d'XMLHttpRequest
- Le développeur moyen est incapable d'ajouter une interface cryptographique propre, et encore moins un protocole correct
- Quelques librairies existent (pour OAuth v1), mais elles sont rares

Outline

Sécurité du réseau

Le Protocole HTTP

Verbes/méthodes

- GET : récupérer le contenu lié à une adresse donnée
- POST : ajout d'une ressource au serveur
- PUT : changement d'une ressource sur un serveur
- DELETE : suppression d'une ressource sur un serveur

Codage des données

- GET : les paramètres sont codés dans l'URL du message :

  `http://www.google.com/webhp?q=ICE`

- POST/PUT : La requête contient un document (XML ou texte) représentant la ressource à créer ou mettre à jour

Suite : chiffrer correctement les documents XML

Le Protocole HTTP

Verbes/méthodes

- GET : récupérer le contenu lié à une adresse donnée
- POST : ajout d'une ressource au serveur
- PUT : changement d'une ressource sur un serveur
- DELETE : suppression d'une ressource sur un serveur

Codage des données

- GET : les paramètres sont codés dans l'URL du message :

```
http://www.google.com/webhp?q=ICE
```

- POST/PUT : La requête contient un document (XML ou texte) représentant la ressource à créer ou mettre à jour

Suite : chiffrer correctement les documents XML

Le Protocole HTTP

Verbes/méthodes

- GET : récupérer le contenu lié à une adresse donnée
- POST : ajout d'une ressource au serveur
- PUT : changement d'une ressource sur un serveur
- DELETE : suppression d'une ressource sur un serveur

Codage des données

- GET : les paramètres sont codés dans l'URL du message :

```
http://www.google.com/webhp?q=ICE
```

- POST/PUT : La requête contient un document (XML ou texte) représentant la ressource à créer ou mettre à jour

Suite : chiffrer correctement les documents XML

CRYPTOGRAPHIE SUR LES DOCUMENTS XML

UTILITÉ

- Pour les services REST (pour les SOAP, autant utiliser les standards)
- Pour les verbes POST/PUT (pour les GET, données dans l'url)
- Lorsqu'on veut un certain niveau d'inter-opérabilité

XML-DSIG

XML Digital Signature

But :

- Définit comment une signature est à appliquer sur un nœud XML
- Les définitions sont réutilisées pour XML-ENC (encryption)

- On ne verra pas les spécifications des algorithmes et clefs de chiffrement (prendre RSA-OAEP, AESxxx ou DSA/RSAwithSHA1 suivant le type de primitive voulu)
- Cf. cours éventuel de cryptographie
- Par contre, il y a des subtilités dues au XML

XML-DSIG

XML Digital Signature

But :

- Définit comment une signature est à appliquer sur un nœud XML
- Les définitions sont réutilisées pour XML-ENC (encryption)

- On ne verra pas les spécifications des algorithmes et clefs de chiffrement (prendre RSA-OAEP, AESxxx ou DSA/RSAwithSHA1 suivant le type de primitive voulu)
- Cf. cours éventuel de cryptographie
- Par contre, il y a des subtilités dues au XML

Rappel :

- On veut signer un arbre XML
- pas sa représentation, qui peut changer au cours des relais
- À partir de SAX/DOM/AXIOM, il faut récréer du texte

CANONICALISATION :

- écriture canonique d'un fichier XML
- plusieurs algorithmes disponibles, il faut en choisir un
- Cet algorithme doit être précisé dans la signature

Subtilités de XML...

Canonicalisation

Rappel :

- On veut signer un arbre XML
- pas sa représentation, qui peut changer au cours des relais
- À partir de SAX/DOM/AXIOM, il faut récréer du texte

Canonicalisation :

- écriture canonique d'un fichier XML
- plusieurs algorithmes disponibles, il faut en choisir un
- Cet algorithme doit être précisé dans la signature

Subtilités de XML...

Canonicalisation

Rappel :

- On veut signer un arbre XML
- pas sa représentation, qui peut changer au cours des relais
- À partir de SAX/DOM/AXIOM, il faut récréer du texte

Canonicalisation :

- écriture canonique d'un fichier XML
- plusieurs algorithmes disponibles, il faut en choisir un
- Cet algorithme doit être précisé dans la signature

SUBTILITÉS DE XML...

- Dans le document :

```
<ns1:a><ns2:b>toto</ns2:b></ns1:a>
```

  Le nom complet du nœud `b` est `ns1:ns2:b`
- L'algorithme normal de canonicalisation commence par recréer le nom complet
- Problème typique : création d'une signature dans le `body` d'un message, qui devient invalide quand on place le `body` dans une `envelope`
- Suivant le type d'application qu'on veut, il faut choisir un algorithme de canonicalisation qui rend la signature portable (en enlevant les ns) ou pas.
- On parle de Exclusive Canonicalisation
- exemple d'utilisation : signature de tokens...
- Alternative : signer le contenu textuel d'un élément, pas l'élément

Subtilités de XML...

- Dans le document :

```
<ns1:a><ns2:b>toto</ns2:b></ns1:a>
```

  Le nom complet du nœud `b` est `ns1:ns2:b`
- L'algorithme normal de canonicalisation commence par recréer le nom complet
- Problème typique : création d'une signature dans le `body` d'un message, qui devient invalide quand on place le `body` dans une `envelope`
- Suivant le type d'application qu'on veut, il faut choisir un algorithme de canonicalisation qui rend la signature portable (en enlevant les ns) ou pas.
- On parle de Exclusive Canonicalisation
- exemple d'utilisation : signature de tokens...
- Alternative : signer le contenu textuel d'un élément, pas l'élément

SUBTILITÉS DE XML...

- Dans le document :

```
<ns1:a><ns2:b>toto</ns2:b></ns1:a>
```

  Le nom complet du nœud `b` est `ns1:ns2:b`
- L'algorithme normal de canonicalisation commence par recréer le nom complet
- Problème typique : création d'une signature dans le `body` d'un message, qui devient invalide quand on place le `body` dans une `envelope`
- Suivant le type d'application qu'on veut, il faut choisir un algorithme de canonicalisation qui rend la signature portable (en enlevant les ns) ou pas.
- On parle de Exclusive Canonicalisation
- exemple d'utilisation : signature de tokens...
- Alternative : signer le contenu textuel d'un élément, pas l'élément

Plan

Un serveur Web en pratique

Serveur non-isolé

- Utilisation par exemple d'Amazon S3 pour stocker les données importantes
- Utilisation d'un STS pour gérer les jetons d'une application professionnelle
- etc. (suite du cours)

Serveur virtuel

- De moins en moins hébergé sur un ordinateur physiquement à disposition
- Cloud : plusieurs sortes plus ou moins virtuelles

Un serveur Web en pratique

Serveur non-isolé

- Utilisation par exemple d'Amazon S3 pour stocker les données importantes
- Utilisation d'un STS pour gérer les jetons d'une application professionnelle
- etc. (suite du cours)

Serveur virtuel

- De moins en moins hébergé sur un ordinateur physiquement à disposition
- Cloud : plusieurs sortes plus ou moins virtuelles

Suite du cours

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser

- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens

- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

SUITE DU COURS

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser

- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens

- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

Suite du cours

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser

- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens

- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

Suite du cours

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser

- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens

- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

SUITE DU COURS

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser
- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens
- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

Suite du cours

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser
- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens
- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

Suite du cours

- Impact de la virtualisation sur la sécurité des serveurs

  Quelques exemples de problèmes pouvant se poser
- Protection contre les utilisateurs malhonnêtes

  Quelques règles de bon sens
- Impact de la non-isolation sur l'architecture des applications sécurisées

  Écriture de politiques de contrôle d'accès réparties entre organisation, application, etc.

Outline

VIRTUALISATION

CONTEXTE D'UTILISATION :

- Infrastructure as a Service (IaaS)
- systèmes d'entreprise complexes

PRINCIPE

- des machines virtuelles (invitées, guest) sont gérées par un système d'exploitation sur une machine réelle (hôte,host)
- La virtualisation peut-être :
  - profonde : l'OS ou l'architecture de la machiné invitée peuvent être différent de celui/celle de la machine hôte
  - superficielle : l'OS et l'architecture de la machine invitée est exactement le même que celui de la machine hôte

VIRTUALISATION

CONTEXTE D'UTILISATION :

- Infrastructure as a Service (IaaS)
- systèmes d'entreprise complexes

PRINCIPE

- des machines virtuelles (invitées, guest) sont gérées par un système d'exploitation sur une machine réelle (hôte,host)
- La virtualisation peut-être :
  - profonde : l'OS ou l'architecture de la machiné invitée peuvent être différent de celui/celle de la machine hôte
  - superficielle : l'OS et l'architecture de la machine invitée est exactement le même que celui de la machine hôte

Exemple : VirtualBox

Fonctionnement de Xen différent, mais sécurité idem

En quelques mots

- logiciel fournit par Oracle (gratuit)
- permet la virtualisation profonde
- abstrait une machine réelle en interceptant tous les appels du système invité

La machine communique *via* une carte réseau (virtuelle par défaut) avec le système hôte et l'extérieur.

Exemple : LXC

En quelques mots

- *Linux containers*
- *Très* inspiré des *zones* de (Sun|Oracle) Solaris
- la machine virtuelle utilise une partie des fichiers systèmes de la machine hôte

La machine communique *via* une carte réseau (virtuelle par défaut) avec le système hôte et l'extérieur.

La machine hôte (jargon Solaris : zone globale) a cependant accès aux fichiers des machines invitées.

Exemple : LXC

En quelques mots

- *Linux containers*
- *Très* inspiré des *zones* de (Sun|Oracle) Solaris
- la machine virtuelle utilise une partie des fichiers systèmes de la machine hôte

La machine communique *via* une carte réseau (virtuelle par défaut) avec le système hôte et l'extérieur.
La machine hôte (jargon Solaris : zone globale) a cependant accès aux fichiers des machines invitées.

Comparaison des modes de virtualisation

En quelques mots :

- La virtualisation profonde implique qu'il faut installer le système invité sur un fichier qui représente le disque dur auquel ce système aura accès
- Il vaut mieux compter quelques dizaines de Go
- En comparaison, le coût d'une virtualisation superficielle avec des containers est de quelques dizaines de Mo
- La virtualisation superficielle se fait avec un minimum de perte de performance

Dans les 2 cas, les machines invitées n'ont pas d'accès direct au contenu non-partagé de la machine hôte. La virtualisation permet donc la ségrégation.

VIRTUALISATION SUPERFICIELLE ET SÉPARATION

COMMENT ÇA MARCHE ?

- Un seul système d'exploitation
- Le noyau de ce système est préparé pour assurer la ségrégation :
  - file descriptors
  - `/dev/`, `/proc/` séparés (copie)
  - `pid` séparés

IMPORTANT

La base de confiance est le noyau du système hôte en entier

VIRTUALISATION SUPERFICIELLE ET SÉPARATION

COMMENT ÇA MARCHE ?

- Un seul système d'exploitation
- Le noyau de ce système est préparé pour assurer la ségrégation :
  - file descriptors
  - `/dev/`, `/proc/` séparés (copie)
  - `pid` séparés

IMPORTANT

La base de confiance est le noyau du système hôte en entier

JAILS/FAKEROOT

BASE DE LA VIRTUALISATION

- historiquement, premières tentatives
- toujours utilisées (pour compiler un noyau, par exemple)
- change le répertoire racine du système de fichier vu par l'application

EXEMPLE

- Création d'un système Linux avec debootstrap
- Utilisation de `chroot` pour créer /dev/ et l'installation de packages supplémentaires dans ce système
- Scripts bash disponible sur demande

JAILS/FAKEROOT

BASE DE LA VIRTUALISATION

- historiquement, premières tentatives
- toujours utilisées (pour compiler un noyau, par exemple)
- change le répertoire racine du système de fichier vu par l'application

EXEMPLE

- Création d'un système Linux avec debootstrap
- Utilisation de `chroot` pour créer /dev/ et l'installation de packages supplémentaires dans ce système
- Scripts bash disponible sur demande

JAIL ESCAPE

IMPORTANT

Dépend des SE, marche pour Linux

SORTIE D'UN ENVIRONNEMENT CHROOTÉ

En étant `root` dans l'environnement, écrire un programme qui :

1. crée un nouveau répertoire et garde un fd sur le répertoire courant
2. fait un `chroot` sur ce nouveau
3. remonte au répertoire parent

COMMENTAIRES

ALAN COX : comportement normal de chroot, qui n'a pas été conçu pour la sécurité.

Un hacker root dans le sous-système a des milliers de façons de devenir root global, de toutes manières.

FREEBSD : hack interdisant l'appel à chroot pour un processus ayant déjà ouvert un répertoire

JAIL ESCAPE

IMPORTANT

Dépend des SE, marche pour Linux

SORTIE D'UN ENVIRONNEMENT CHROOTÉ

En étant `root` dans l'environnement, écrire un programme qui :

1. crée un nouveau répertoire et garde un fd sur le répertoire courant
2. fait un `chroot` sur ce nouveau
3. remonte au répertoire parent

COMMENTAIRES

ALAN COX : comportement normal de chroot, qui n'a pas été conçu pour la sécurité.

Un hacker root dans le sous-système a des milliers de façons de devenir root global, de toutes manières.

FREEBSD : hack interdisant l'appel à chroot pour un processus ayant déjà ouvert un répertoire

JAIL ESCAPE

IMPORTANT

Dépend des SE, marche pour Linux

SORTIE D'UN ENVIRONNEMENT CHROOTÉ

En étant `root` dans l'environnement, écrire un programme qui :

1. crée un nouveau répertoire et garde un fd sur le répertoire courant
2. fait un `chroot` sur ce nouveau
3. remonte au répertoire parent

COMMENTAIRES

ALAN COX : comportement normal de chroot, qui n'a pas été conçu pour la sécurité.

Un hacker root dans le sous-système a des milliers de façons de devenir root global, de toutes manières.

FREEBSD : hack interdisant l'appel à chroot pour un processus ayant déjà ouvert un répertoire

USERS ID

POUR LXC UNIQUEMENT :

- le SE ne sépare pas les uid dans les environnements chrootés ou pas
- utilisateur 100 dans l'un = utilisateur 100 à l'extérieur
- Encore un problème de violation de partitionnement

IMPORTANT

Mais pas encore d'exploitation triviale de ce problème

USERS ID

POUR LXC UNIQUEMENT :

- le SE ne sépare pas les uid dans les environnements chrootés ou pas
- utilisateur 100 dans l'un = utilisateur 100 à l'extérieur
- Encore un problème de violation de partitionnement

IMPORTANT

Mais pas encore d'exploitation triviale de ce problème

CAPACITÉS

EN AVANCE SUR LE COURS

CAPABILITIES

- Mandatory Access Control sous linux
- Une capability correspond à un ensemble de droits (*e.g.*, accès carte réseau, accès système de fichiers,. . .)
- Permet d'être beaucoup plus fin que root/pas root

SOLUTION IMMATURE

- Solution trop récente
- Les développeurs réutilisent toujours les mêmes capacités
- En pratique, `CAP_SYS_ADMIN` est utilisé dans 30% des cas

  Avoir cette capacité est plus ou moins être root
- En progrès ?

CAPACITÉS

EN AVANCE SUR LE COURS

CAPABILITIES

- Mandatory Access Control sous linux
- Une capability correspond à un ensemble de droits (*e.g.*, accès carte réseau, accès système de fichiers,. . .)
- Permet d'être beaucoup plus fin que root/pas root

SOLUTION IMMATURE

- Solution trop récente
- Les développeurs réutilisent toujours les mêmes capacités
- En pratique, `CAP_SYS_ADMIN` est utilisé dans 30% des cas

  Avoir cette capacité est plus ou moins être root
- En progrès ?

Quel type de virtualisation mon hébergeur utilise-t-il ?

- D'abord il faut se loguer sur l'hébergeur (voir le manuel)
- Différence typique : les systèmes de fichiers ne sont pas montés de l'intérieur du container
- Donc pas de /etc/mtab
- On peut aussi regarder `/proc/mount`, pour voir les fichiers montés individuellement

Important

Essayer de se renseigner avant !

Outline

Architecture des serveurs Web

- Virtualisation
- Architectures d'applications Web
  - Services ReST
  - Services SOAP

PRINCIPE DE BASE

APPLICATIONS

- contrôle de flot : if. . . , while. . . , etc.
- état : valeur des variables manipulées par le programme
- = les C et M de MVC (autre cours)

CLIENT/SERVEUR SANS ÉTAT

- Le serveur fournit des méthodes d'accès à une base de données

  modèle sur lequel les opérations sont effectués
- Le client gère la partie contrôle de l'application
- Le client doit aussi gérer la partie de l'état servant au contrôle

AVANTAGES

- Gestion simplifiée du serveur
- Basé sur les *verbes* de HTTP pour décrire le mode d'interaction avec des requêtes avec les données (put, get, update, delete)
- Facile à mettre en œuvre (JavaScript/JSON(P) et python, ruby, . . .)

INCONVÉNIENT

- C'est un principe, pas un standard (programmation objet *vs.* Java)

  Le format des messages n'est pas uniforme
- Difficile à expliquer de manière uniforme

Conseil : jouer un peu avec Ruby on Rails

Simple Object Access Protocol

Mais maintenant, seulement SOAP !

Architecture en couche

Pour les réseaux, architecture en couche classique :

- Une couche fournit des services à une couche supérieure
- Pour fonctionner, elle a besoin de services rendus par une couche inférieure

- Dans tout le reste du cours, cette architecture en couche est fondamentale
- Le *Binding* est ce qui relie la couche qu'on est en train de définir/d'étudier/d'utiliser à la couche inférieure
- SOAP est un protocole défini dans la couche applicative

Architecture en couche

Pour les réseaux, architecture en couche classique :

- Une couche fournit des services à une couche supérieure
- Pour fonctionner, elle a besoin de services rendus par une couche inférieure

- Dans tout le reste du cours, cette architecture en couche est fondamentale
- Le *Binding* est ce qui relie la couche qu'on est en train de définir/d'étudier/d'utiliser à la couche inférieure
- SOAP est un protocole défini dans la couche applicative

SOAP NODE

- L'émetteur d'un message
- Son receveur ultime
- Ou un relais, qui fera éventuellement des opérations sur le message avant de le rediriger

FORMAT D'UN MESSAGE

- Un message est un document XML
- Le nœud racine est `envelope`
- Ce nœud a pour fils 0 ou un nœud `header`, qui a pour fils 0 ou plusieurs nœuds (*header block*) qui sont utilisés par les SOAP Node pour déterminer ce qu'ils doivent faire du message
- Le nœud `envelope` a aussi un unique fils `body`, qui est traité par le receveur ultime du message

CONCEPTS DE BASE

HTTP://WWW.W3.ORG/TR/SOAP12-PART1/

SOAP NODE

- L'émetteur d'un message
- Son receveur ultime
- Ou un relais, qui fera éventuellement des opérations sur le message avant de le rediriger

FORMAT D'UN MESSAGE

- Un message est un document XML
- Le nœud racine est `envelope`
- Ce nœud a pour fils 0 ou un nœud `header`, qui a pour fils 0 ou plusieurs nœuds (*header block*) qui sont utilisés par les SOAP Node pour déterminer ce qu'ils doivent faire du message
- Le nœud `envelope` a aussi un unique fils `body`, qui est traité par le receveur ultime du message

Exemple

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
    <n:priority>1</n:priority>
    <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

On s'intéresse surtout au cas simple, un header, un body

Message Exchange Patterns

http://www.w3.org/TR/soap12-part2/#soapsupmep

- Définition de conversations entre un client et un serveur : ex : request/response
- Statut du message dans cette conversion : ex : Inbound/outbound
- Informations supplémentaires : immediateSender, immediateReceiver
- But : faciliter le traitement du message,
- Aucun impact sur la sécurité, ajouté ici pour pouvoir ensuite parler du traitement des messages

Bindings

http://www.w3.org/TR/soap12-part2/#soapinhttp

- SOAP définit un message comme un *InfoSet* XML
- Un *Binding* définit comment cet InfoSet peut être représenté comme *charge* dans un protocole de transport
- HTTP : Un *binding* standard, mais qui n'est pas obligatoire, mais naturel pour des documents XML
- Note pour la sécurité : c'est la destination indiquée dans le message SOAP qui est pertinente pour son traitement, pas celle indiquée dans le header du message HTTP
- c'est une source connue de failles de sécurité
- Le verbe HTTP ne peut être que get (inbound) ou post (outbound)
- Binding alternatifs : e-mail, JMS, etc.
- HTTP définit lui-même des bindings pour le transport : IP, SSL (TLS)

Exemple SOAP sur HTTP

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

CE QU'IL FAUT RETENIR...

- SOAP est un protocole "logique" qui s'intéresse à l'InfoSet d'un document XML plus qu'à son texte
- Les intermédiaires peuvent changer le texte du document en :
  - enlevant des morceaux de header qui leur étaient destinés
  - ajoutant/enlevant des *white space*
  - ajout des commentaires, ou de nouveaux headers
- Certaines interactions mettent en jeu plusieurs entités

WS Description Language

http://www.w3.org/TR/wsdl

Rôle :

- Description de l'interface (*contrat*) permettant d'utiliser un service
- Analogie : le *.h* d'une librairie C

Doit définir :

- Les rôles (*portType*) possibles que joue le serveur
- Les *opérations* disponibles dans chaque rôle
- Le type de chaque opération (*e.g.* consomme ou initie une requête)
- Le format des messages (*message*) pour ces opérations
- Un *binding* qui spécifie, pour chaque opération, le protocole à utiliser
- Un *port* qui spécifie l'adresse (*endpoint*) permettant d'appeler l'opération

Un *service* est un ensemble de ports

ABSTRAIT/CONCRET

Dans ce que définit un fichier WSDL, il y a une partie abstraite et un concrète.

PARTIE ABSTRAITE

- rôle
- portType
- opérations
- message

ABSTRAIT/CONCRET

Dans ce que définit un fichier WSDL, il y a une partie abstraite et un concrète.

PARTIE CONCRÈTE

- binding, qui définit :
  - le protocole (*e.g.* SOAP avec binding HTTP)
  - comment le message est encodé dans un message du protocole :
    - RPC literal/encoded (= avec information de typage)
    - Document literal (sans le nom de l'opération, pas interopérable)/literal wrapped (avec)
- port, service

Axis2

- Utilisable en stand-alone ou derrière Apache
- Permet de déployer très rapidement du code Java :
  - génération automatisée du WSDL
  - changement à chaud
- Fonctionnement basé sur :
  - un flot de traitement sur un message
  - un état global qui est mis à jour par chaque filtre dans la flot
  - à la fin du flot, la classe et la méthode Java à appeler sont retrouvées dans l'état global (anc. pivot)
  - puis réponse en faisant le chemin dans l'autre sens

Utilisation de Axis2 pour la sécurité

- Insertion dans le flot du module `Rampart` (fichier axis2.xml), automatique lors de l'installation
- Reprendre le fichier `service.xml` généré automatiquement, et y ajouter un nœud `wsp:policy` qui contient une politique de sécurité WS-SecurityPolicy
- Cette politique est ensuite mappée automatiquement sur le WSDL

Note :

L'ancienne méthode consistait à annoter le WSDL à la main, aux différents endroits qui conviennent (dans les bindings) avec des références vers une `policy`

Plan

THE SHOW SO FAR...

- On a vu, très rapidement, comment utiliser de la cryptographie dans les messages échangés entre services
- Partie importante : utilisation de tokens pour valider l'accès à un service
- Au-delà de l'aspect technique,

  comment écrire correctement des politiques de contrôle d'accès ?

  Question annexe : que veut/doit-on protéger ?
- À quel endroit faire les vérifications pour le contrôle d'accès ?

THE SHOW SO FAR...

- On a vu, très rapidement, comment utiliser de la cryptographie dans les messages échangés entre services
- Partie importante : utilisation de tokens pour valider l'accès à un service
- Au-delà de l'aspect technique,

  comment écrire correctement des politiques de contrôle d'accès ?

  Question annexe : que veut/doit-on protéger ?
- À quel endroit faire les vérifications pour le contrôle d'accès ?

Outline

Contrôle d'accès (théorie)

- Base logique
- Classification
- Modèle Bell-LaPadula
- Modèle Biba
- Le modèle Harrisson-Ruzzo-Ullman
- Role-Based Access Control (Généralités)

PRÉ-REQUIS

BASES DE DONNÉES

- savoir écrire une requête (de base)
- savoir ce qu'est un schéma
- savoir ce qu'est une entrée dans une table

et savoir renommer les choses

noms savants : Clauses de Horn et DATALOG

PRÉ-REQUIS

BASES DE DONNÉES

- savoir écrire une requête (de base)
- savoir ce qu'est un schéma
- savoir ce qu'est une entrée dans une table

et savoir renommer les choses

noms savants : Clauses de Horn et DATALOG

PRÉ-REQUIS

BASES DE DONNÉES

- savoir écrire une requête (de base)
- savoir ce qu'est un schéma
- savoir ce qu'est une entrée dans une table

et savoir renommer les choses

noms savants : Clauses de Horn et DATALOG

Algèbre

- Constantes : ce sont des valeurs dans un type donné (qui ne nous regarde pas)

  Exemple : "Catherine ", 5.67, -1,. . .

- Variables : . . . . Rappel : elles sont utilisées dans les requêtes, pas dans les entrées
- termes : ce sont des expressions construites à partir des constantes et des variables.

  Exemple : "Catherine "^"Henry ", 5.67 $\times$ x,. . .

Algèbre

- Constantes : ce sont des valeurs dans un type donné (qui ne nous regarde pas)

  Exemple : "Catherine ", 5.67, -1,. . .

- Variables : . . . . Rappel : elles sont utilisées dans les requêtes, pas dans les entrées
- termes : ce sont des expressions construites à partir des constantes et des variables.

  Exemple : "Catherine "^"Henry ", $5.67 \times x$,. . .

Algèbre

- Constantes : ce sont des valeurs dans un type donné (qui ne nous regarde pas)

  Exemple : "Catherine ", 5.67, -1,. . .

- Variables : . . . . Rappel : elles sont utilisées dans les requêtes, pas dans les entrées
- termes : ce sont des expressions construites à partir des constantes et des variables.

  Exemple : "Catherine "^"Henry ", $5.67 \times x$,. . .

Algèbre

- Constantes : ce sont des valeurs dans un type donné (qui ne nous regarde pas)

  Exemple : "Catherine ", 5.67, -1,. . .

- Variables : . . . . Rappel : elles sont utilisées dans les requêtes, pas dans les entrées
- termes : ce sont des expressions construites à partir des constantes et des variables.

  Exemple : "Catherine "^"Henry ", 5.67 $\times$ x,. . .

Algèbre

- Constantes : ce sont des valeurs dans un type donné (qui ne nous regarde pas)

  Exemple : "Catherine ", 5.67, -1,. . .

- Variables : . . . . Rappel : elles sont utilisées dans les requêtes, pas dans les entrées
- termes : ce sont des expressions construites à partir des constantes et des variables.

  Exemple : "Catherine "^"Henry ", $5.67 \times x$,. . .

Algèbre

- Constantes : ce sont des valeurs dans un type donné (qui ne nous regarde pas)

  Exemple : "Catherine ", 5.67, -1,. . .

- Variables : . . . . Rappel : elles sont utilisées dans les requêtes, pas dans les entrées
- termes : ce sont des expressions construites à partir des constantes et des variables.

  Exemple : "Catherine "^"Henry ", $5.67 \times x$,. . .

Relations

- Symboles dont les arguments sont des termes
- Fondamental en logique !
- Exemple 1 : schémas dans une base de données
- Exemple 2 : arcs dans un graphe
- On s'intéresse plus à la première notion

Relations

- Symboles dont les arguments sont des termes
- Fondamental en logique !
- Exemple 1 : schémas dans une base de données
- Exemple 2 : arcs dans un graphe
- On s'intéresse plus à la première notion

Relations

- Symboles dont les arguments sont des termes
- Fondamental en logique !
- Exemple 1 : schémas dans une base de données
- Exemple 2 : arcs dans un graphe
- On s'intéresse plus à la première notion

Relations

- Symboles dont les arguments sont des termes
- Fondamental en logique !
- Exemple 1 : schémas dans une base de données
- Exemple 2 : arcs dans un graphe
- On s'intéresse plus à la première notion

Relations

- Symboles dont les arguments sont des termes
- Fondamental en logique !
- Exemple 1 : schémas dans une base de données
- Exemple 2 : arcs dans un graphe
- On s'intéresse plus à la première notion

Relations

- Symboles dont les arguments sont des termes
- Fondamental en logique !
- Exemple 1 : schémas dans une base de données
- Exemple 2 : arcs dans un graphe
- On s'intéresse plus à la première notion

Interprétation

- on interprète les termes dans un domaine $\mathscr{D}$

  Exemple : $s(s(0))$ est interprété comme l'entier 2

- on interprète une relation comme une relation sur le domaine

  Exemple : $pp(x, y)$ est interprété comme $x < y$ avec x, y entiers

- Une relation est vraie dans une interprétation si sa traduction est vraie dans le domaine

Interprétation

- on interprète les termes dans un domaine $\mathscr{D}$

  Exemple : $s(s(0))$ est interprété comme l'entier 2

- on interprète une relation comme une relation sur le domaine

  Exemple : $pp(x, y)$ est interprété comme $x < y$ avec x, y entiers

- Une relation est vraie dans une interprétation si sa traduction est vraie dans le domaine

Interprétation

- on interprète les termes dans un domaine $\mathscr{D}$

  Exemple : $s(s(0))$ est interprété comme l'entier 2

- on interprète une relation comme une relation sur le domaine

  Exemple : $pp(x, y)$ est interprété comme $x < y$ avec x, y entiers

- Une relation est vraie dans une interprétation si sa traduction est vraie dans le domaine

INTERPRÉTATION

- on interprète les termes dans un domaine $\mathscr{D}$

  Exemple : $s(s(0))$ est interprété comme l'entier 2

- on interprète une relation comme une relation sur le domaine

  Exemple : $pp(x, y)$ est interprété comme $x < y$ avec x, y entiers

- Une relation est vraie dans une interprétation si sa traduction est vraie dans le domaine

Interprétation

- on interprète les termes dans un domaine $\mathscr{D}$

  Exemple : $s(s(0))$ est interprété comme l'entier 2

- on interprète une relation comme une relation sur le domaine

  Exemple : $pp(x, y)$ est interprété comme $x < y$ avec x, y entiers

- Une relation est vraie dans une interprétation si sa traduction est vraie dans le domaine

Interprétation

- on interprète les termes dans un domaine $\mathscr{D}$

  Exemple : $s(s(0))$ est interprété comme l'entier 2

- on interprète une relation comme une relation sur le domaine

  Exemple : $pp(x, y)$ est interprété comme $x < y$ avec x, y entiers

- Une relation est vraie dans une interprétation si sa traduction est vraie dans le domaine

Utilisation des relations en logique

Exercice : Traduire en formules logiques en utilisant des relations

1. Tout batiment d'au moins 4 étages doit avoir un ascenseur, sauf s'il a été construit avant 1945
2. Tout homme est mortel, et Socrate est un homme. Donc Socrate est mortel
3. Tout orage est accompagné de pluie ou d'éclairs. Il ne pleut pas, donc il n'y a pas d'orage.
4. Le barbier de ce village rase exactement tous les hommes qui ne se rasent pas eux-mêmes.
5. Idem, et en plus le barbier s'appelle Pierre.

Dire quels ensembles de clauses sont vraies !

Utilisation des relations en logique

Exercice : Traduire en formules logiques en utilisant des relations

1. Tout batiment d'au moins 4 étages doit avoir un ascenseur, sauf s'il a été construit avant 1945
2. Tout homme est mortel, et Socrate est un homme. Donc Socrate est mortel
3. Tout orage est accompagné de pluie ou d'éclairs. Il ne pleut pas, donc il n'y a pas d'orage.
4. Le barbier de ce village rase exactement tous les hommes qui ne se rasent pas eux-mêmes.
5. Idem, et en plus le barbier s'appelle Pierre.

Dire quels ensembles de clauses sont vraies !

UN DOMAINE SPÉCIAL

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

MORALE

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Un domaine spécial

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

Morale

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Un domaine spécial

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

Morale

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Un domaine spécial

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

Morale

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Un domaine spécial

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

Morale

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Un domaine spécial

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

Morale

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Un domaine spécial

Pour une relation $R(x_1, \ldots, x_n)$

- On considère le domaine D de tous les termes
- Une interprétation de R est un sous-ensemble E de D^n
- $R(t_1, \ldots, t_n)$ est vrai si $(t_1, \ldots, t_n)$ est dans E
- Mais c'est déjà bien connu !
- Une relation, c'est un schéma dans une base de données
- Les entrées d'une table sont les éléments de E

Morale

Le contenu d'une base de données représente les valeurs pour lesquelles le schéma est "vrai"

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

$$R(Nom, Prénom, commune)$$

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

$$R(\text{Nom, Prénom, commune})$$

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

$$R(\text{Nom, Prénom, commune})$$

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

$$R(\text{Nom, Prénom, commune})$$

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

$$R(\text{Nom, Prénom, commune})$$

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

$$R(\text{Nom, Prénom, commune})$$

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Requêtes SQL et logique

Un exemple : le Join

- SELECT Nom, Prénom, commune,
- FROM Communes INNER JOIN Personnes
- ON Communes.code_commune = Personnes.Code_commune ;

Logiquement parlant...

- **Communes(code_commune,commune)**
- **Personnes(Nom, Prénom,code_commune)**
- La commande crée une nouvelle relation R :

  R(Nom, Prénom, commune)

- La requête consiste à rendre vraie la formule logique

$$\forall x, y, z, t, \text{Communes}(x, y) \wedge \text{Personnes}(z, t, x) \Rightarrow R(z, t, x)$$

Clauses de Horn

Notation

$$R(\vec{x}) \leftarrow R_1(\vec{y_1}), \ldots, R_n(\vec{y_n})$$

avec $\vec{x}, \vec{y_1}, \ldots, \vec{y_n}$ des vecteurs de termes

Signification

Pour toute instance des variables de la clause :

- Si toutes les relations à droite de la flêche sont vraies...
- alors la relation à gauche de la règle doit être vraie

CLAUSES DE HORN

NOTATION

$$R(\vec{x}) \leftarrow R_1(\vec{y_1}), \ldots, R_n(\vec{y_n})$$

avec $\vec{x}, \vec{y_1}, \ldots, \vec{y_n}$ des vecteurs de termes

SIGNIFICATION

Pour toute instance des variables de la clause :

- Si toutes les relations à droite de la flêche sont vraies...
- alors la relation à gauche de la règle doit être vraie

Clauses de Horn

Notation

$$R(\vec{x}) \leftarrow R_1(\vec{y_1}), \ldots, R_n(\vec{y_n})$$

avec $\vec{x}, \vec{y_1}, \ldots, \vec{y_n}$ des vecteurs de termes

Signification

Pour toute instance des variables de la clause :

- Si toutes les relations à droite de la flêche sont vraies...
- alors la relation à gauche de la règle doit être vraie

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Différence SQL/Clauses de Horn

Réduction de SQL en Clauses de Horn

- Créer une nouvelle relation pour chaque requête SQL
- Cette relation sera la tête de la clause
- Mettre chaque relation utilisée par la requête à droite
- instancier (ou rendre égales) les variables en fonction de la requête
- Marche tant qu'il n'y a pas de *bag*

Réduction de Clauses de Horn en requêtes SQL

- Marche seulement si on a un ordre sur les clauses tel que la tête d'une clause n'a jamais été utilisée dans le corps d'une clause précédente
- Le but est d'empêcher la récursion
- Récursion explicite autorisée dans SQL 3

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet. . .
- Mais polynomial (P-complet) pour les cas pratiques

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet...
- Mais polynomial (P-complet) pour les cas pratiques

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet...
- Mais polynomial (P-complet) pour les cas pratiques

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet...
- Mais polynomial (P-complet) pour les cas pratiques

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet...
- Mais polynomial (P-complet) pour les cas pratiques

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet...
- Mais polynomial (P-complet) pour les cas pratiques

Prolog et Datalog

Définition

- Dans les 2 cas, on a un ensemble de clauses de Horn
- Pour Prolog, on a aussi des priorités sur les règles
- Pour Datalog, on n'autorise des termes qui sont seulement des variables ou des constantes
- note : règles $R(\vec{x})$ pour désigner des faits

Décidabilité

- Formellement, de savoir si un ensemble de clauses est inconsistant
- Prolog : indécidable.
- Datalog : EXPTIME-complet...
- Mais polynomial (P-complet) pour les cas pratiques

Calcul de point fixe (exemple)

On part d'un ensemble de faits $F = \emptyset$.

Tant qu'on n'a pas finit :

- pour toutes les règles
- regarder si une instance σ du corps d'une règle est dans F, ajouter l'instance σ de la tête à F
- recommencer

Example

Exemple : calcul de la clôture transitive d'une relation

$$\begin{cases} \texttt{agent}(\mathit{alice}), \texttt{agent}(\mathit{bob}), \texttt{agent}(\mathit{charlie}) \\ \texttt{ami}(\mathit{alice}, \mathit{bob}), \texttt{ami}(\mathit{bob}, \mathit{charlie}) \\ \texttt{ami}(X, Y) \leftarrow \texttt{ami}(X, Z), \texttt{ami}(Z, Y) \end{cases}$$

Calcul de point fixe (exemple)

On part d'un ensemble de faits $F = \emptyset$.

Tant qu'on n'a pas finit :

- pour toutes les règles
- regarder si une instance σ du corps d'une règle est dans F, ajouter l'instance σ de la tête à F
- recommencer

Example

Exemple : calcul de la clôture transitive d'une relation

$$\begin{cases} \texttt{agent}(\mathit{alice}), \texttt{agent}(\mathit{bob}), \texttt{agent}(\mathit{charlie}) \\ \texttt{ami}(\mathit{alice}, \mathit{bob}), \texttt{ami}(\mathit{bob}, \mathit{charlie}) \\ \texttt{ami}(X, Y) \leftarrow \texttt{ami}(X, Z), \texttt{ami}(Z, Y) \end{cases}$$

Calcul de point fixe (exemple)

On part d'un ensemble de faits $F = \emptyset$.

Tant qu'on n'a pas finit :

- pour toutes les règles
- regarder si une instance σ du corps d'une règle est dans F, ajouter l'instance σ de la tête à F
- recommencer

Example

Exemple : calcul de la clôture transitive d'une relation

$$\begin{cases} \texttt{agent}(\mathit{alice}), \texttt{agent}(\mathit{bob}), \texttt{agent}(\mathit{charlie}) \\ \texttt{ami}(\mathit{alice}, \mathit{bob}), \texttt{ami}(\mathit{bob}, \mathit{charlie}) \\ \texttt{ami}(X, Y) \leftarrow \texttt{ami}(X, Z), \texttt{ami}(Z, Y) \end{cases}$$

Calcul de point fixe (exemple)

On part d'un ensemble de faits $F = \emptyset$.
Tant qu'on n'a pas finit :

- pour toutes les règles
- regarder si une instance σ du corps d'une règle est dans F, ajouter l'instance σ de la tête à F
- recommencer

Example

Exemple : calcul de la clôture transitive d'une relation

$$\begin{cases} \texttt{agent}(\mathit{alice}), \texttt{agent}(\mathit{bob}), \texttt{agent}(\mathit{charlie}) \\ \texttt{ami}(\mathit{alice}, \mathit{bob}), \texttt{ami}(\mathit{bob}, \mathit{charlie}) \\ \texttt{ami}(X, Y) \leftarrow \texttt{ami}(X, Z), \texttt{ami}(Z, Y) \end{cases}$$

Outline

Contrôle d'accès (théorie)

Types de contrôle d'accès (1/2)

Unix|Windows

- un utilisateur est propriétaire des fichiers
- c'est à l'utilisateur de choisir les droits d'accès aux fichiers qu'il possède (y compris donner le fichier à un autre utilisateur)

chmod, chown,...

Types de contrôle d'accès (2/2)

Systèmes d'entreprises

- un utilisateur travaille sur des données qui ne sont pas *à lui*

  exemple : manipulation d'une fiche client
- les lois obligent à respecter certaines conditions pour le contrôle d'accès
- les utilisateurs ne sont pas forcément compétents

  $\Rightarrow$ Un *administrateur* fixe les droits d'accès aux fichiers

On parle de contrôle d'accès discrétionnaire (laissé à la *discrétion* de l'utilisateur) ou obligatoire (*Mandatory*–MAC, fixé par une autorité)

MAC POUR LES GRANDS SYSTÈMES

ACL – ACCESS CONTROL LISTS

$$ACL : \text{object} \rightarrow (\ \text{user} \times \text{action list}\ )\ \text{list}$$

- Un fichier est un *objet*
- L'administrateur attache à chaque objet une liste contenant :
  - les noms de tous les utilisateurs pouvant accéder à l'objet
  - Pour chaque utilisateur, la liste des actions permises (lecture, etc.)

EN PRATIQUE

- Utilisateurs ou *groupes* d'utilisateurs
- implémenté dans SELinux, Solaris et Oracle Linux, FreeBSD, ...
- fichier au sens Unix : Pour les exécutables, possibilité de restreindre les appels systèmes (suivant implémentation)

MAC POUR LES GRANDS SYSTÈMES

ACL – ACCESS CONTROL LISTS

$$ACL : \text{object} \rightarrow (\ \text{user} \times \text{action list}\ )\ \text{list}$$

- Un fichier est un *objet*
- L'administrateur attache à chaque objet une liste contenant :
  - les noms de tous les utilisateurs pouvant accéder à l'objet
  - Pour chaque utilisateur, la liste des actions permises (lecture, etc.)

EN PRATIQUE

- Utilisateurs ou *groupes* d'utilisateurs
- implémenté dans SELinux, Solaris et Oracle Linux, FreeBSD, ...
- fichier au sens Unix : Pour les exécutables, possibilité de restreindre les appels systèmes (suivant implémentation)

Outline

Contrôle d'accès (théorie)

Origines

Contexte (1975)

- Début des systèmes multi-tâches, multi-utilisateurs
- Développé au MITRE pour des systèmes de haute-sécurité
- Sert de base au *Trusted Computer System Evaluation Criteria (TCSEC)* (DoD, Orange book)
- Basé sur une série d'articles définissant ce qu'est la sécurité

Intérêt de son étude :

- Défense, aéronautique
- Dans l'esprit NEAT
- Étalon pour mesurer la sécurité dans d'autres systèmes

Origines

Contexte (1975)

- Début des systèmes multi-tâches, multi-utilisateurs
- Développé au MITRE pour des systèmes de haute-sécurité
- Sert de base au *Trusted Computer System Evaluation Criteria (TCSEC)* (DoD, Orange book)
- Basé sur une série d'articles définissant ce qu'est la sécurité

Intérêt de son étude :

- Défense, aéronautique
- Dans l'esprit NEAT
- Étalon pour mesurer la sécurité dans d'autres systèmes

Bases

Sujets et objets

- Un utilisateur, ou un programme incarnant l'utilisateur, est un *sujet*
- Un fichier ou une ressource est un *objet*
- (Suivant le niveau d'abstraction) : tout sujet est un objet

Environnement "militaire"

- Le contrôle d'accès est exprimé par une une autorité
- Cette autorité a plus ou moins confiance en les sujets pour préserver la confidentialité des données : habilitation/*clearance level*
- Cette autorité veille à ce que ses données ne soient accessibles qu'aux sujets habilités : niveau de sécurité

Hiérarchie de base

En France

Très Secret Défense > Secret Défense > Confidentiel Défense > informations protégées (autre hiérarchie)

- Applicable aux industries connexes
- Gros projets industriels :
  - Restriction sur la diffusion des données (Besoin de Savoir)
  - Hiérarchie similaire applicable au sein d'un projet
- Dépendances entre projets : possibilité de diffuser de l'information à certains niveaux entre projets.

TREILLIS DES NIVEAUX DE SÉCURITÉ

DÉFINITION PAS FORMELLE

- Dans un arbre, 2 nœuds ont toujours un ancêtre commun minimal
- Dans un treillis, en plus, deux nœuds ont toujours un descendant commun

- À partir de n'importe quel ordre, ajouter un élément maximum et un minimum permet d'obtenir un treillis
- Exemple : public (minimum), TSD+ (maximum)

CURIOSITÉ

SOUS-BUT DU SYSTÈME DE CONTRÔLE D'ACCÈS :

Un sujet qui a une habilitation L_s ne peut lire que des données qui sont à un niveau de sécurité $L_d \leq L_s$.

- No Read UP
- Un sujet curieux ne peut pas ouvrir un dossier s'il n'a pas le niveau d'habilitation suffisant

FUITES

SOUS-BUT DU SYSTÈME DE CONTRÔLE D'ACCÈS :

Un sujet qui a une habilitation L_s ne peut écrire que des données qui sont à un niveau de sécurité $L_d \geq L_s$.

- No Write Down
- Un sujet ne peut pas divulguer le contenu d'un dossier en le plaçant dans un dossier à un niveau inférieur
- Raisonnement en 2 temps :
  - Si c'est possible, alors écriture d'une information SD dans une clef USB (publique)
  - Lecture de la clef USB par n'importe qui (autorisée)
- autre nom : propriété $\star$

Modèle logique

Prédicats

- $\texttt{hab}(x, l)$: le sujet x a l'habilitation l
- $\texttt{niv}(x, l)$: l'objet x a le niveau de sécurité l
- $\texttt{lire}(x, y)$: le sujet x peut lire l'objet y
- $\texttt{ecr}(x, y)$: le sujet x peut écrire dans l'objet y
- $l \preceq l'$: le niveau l est inférieur au niveau l'

Règles de contrôle d'accès

$$\begin{aligned} \texttt{lire}(x, y) &\leftarrow \texttt{hab}(x, l), \texttt{niv}(y, l'), l' \preceq l \\ \texttt{ecr}(x, y) &\leftarrow \texttt{hab}(x, l), \texttt{niv}(y, l'), l \preceq l' \\ l \preceq l' &\leftarrow l \preceq l'', l'' \preceq l' \end{aligned}$$

dernière règle : car l'ordre est transitif

Modèle logique

Prédicats

- $\texttt{hab}(x, l)$: le sujet x a l'habilitation l
- $\texttt{niv}(x, l)$: l'objet x a le niveau de sécurité l
- $\texttt{lire}(x, y)$: le sujet x peut lire l'objet y
- $\texttt{ecr}(x, y)$: le sujet x peut écrire dans l'objet y
- $l \preceq l'$: le niveau l est inférieur au niveau l'

Règles de contrôle d'accès

$$\begin{aligned} \texttt{lire}(x, y) &\leftarrow \texttt{hab}(x, l), \texttt{niv}(y, l'), l' \preceq l \\ \texttt{ecr}(x, y) &\leftarrow \texttt{hab}(x, l), \texttt{niv}(y, l'), l \preceq l' \\ l \preceq l' &\leftarrow l \preceq l'', l'' \preceq l' \end{aligned}$$

dernière règle : car l'ordre est transitif

Modèle logique

Prédicats

- $\texttt{hab}(x, l)$: le sujet x a l'habilitation l
- $\texttt{niv}(x, l)$: l'objet x a le niveau de sécurité l
- $\texttt{lire}(x, y)$: le sujet x peut lire l'objet y
- $\texttt{ecr}(x, y)$: le sujet x peut écrire dans l'objet y
- $l \preceq l'$: le niveau l est inférieur au niveau l'

Règles de contrôle d'accès

$$\begin{aligned} \texttt{lire}(x, y) &\leftarrow \texttt{hab}(x, l), \texttt{niv}(y, l'), l' \preceq l \\ \texttt{ecr}(x, y) &\leftarrow \texttt{hab}(x, l), \texttt{niv}(y, l'), l \preceq l' \\ l \preceq l' &\leftarrow l \preceq l'', l'' \preceq l' \end{aligned}$$

dernière règle : car l'ordre est transitif

Exercice d'application

Avec les règles précédentes, et :

- $a \preceq b \preceq d$ et $a \preceq c \preceq d$
- $\texttt{hab}(s, c)$

Quels sont les niveaux auxquels le sujet s peut lire ? écrire ?

À savoir :

- Calculer si un sujet peut lire/écrire un objet donné
- Calculer les niveaux auquel un sujet peut lire ou écrire
- Calculer les habilitations auxquelles un objet peut être lu ou écrit

EXERCICE D'APPLICATION

Avec les règles précédentes, et :

- $a \preceq b \preceq d$ et $a \preceq c \preceq d$
- $\texttt{hab}(s, c)$

Quels sont les niveaux auxquels le sujet s peut lire ? écrire ?

À SAVOIR :

- Calculer si un sujet peut lire/écrire un objet donné
- Calculer les niveaux auquel un sujet peut lire ou écrire
- Calculer les habilitations auxquelles un objet peut être lu ou écrit

COMMUNICATIONS DANS BLP

COMMUNICATIONS RIGIDES

- Une communication de s vers s' signifie que s écrit un message et que s' le lit
- NRU+NWD : le niveau d'habilitation de s' doit être supérieur à celui de s

Implication : Un directeur ne pourra pas communiquer un message à des sous-traitants !

Classification/Déclassification

Déclassification

- Il faut identifier un sous-ensemble de Trusted Users qui peuvent écrire vers le bas
- La manière dont ces sujets sont choisis n'est pas spécifiée
- Les contraintes (déclaration préalable, log, etc.) auxquelles sont soumises les déclassifications ne sont pas spécifiées

Le système modélise les accès informatiques. Des procédures spéciales sont mises en place pour les humains.

Outline

Contrôle d'accès (théorie)

Origines

Contexte

- Postérieur au modèle Bell-LaPadula
- Problème identifié :
  - Lorsqu'on met des niveaux de sécurité dans une arborescence de fichiers
  - Est-ce que le niveau de sécurité doit être croissant ou décroissant ?

Réponse :

- Décroissant : plus on s'enfonce dans l'arborescence, moins on a besoin de sécurité (comparaison / et /tmp)
- Croissant : car si on ne peut pas connaître un dossier, on ne peut rien savoir de son contenu

ORIGINES

CONTEXTE

- Postérieur au modèle Bell-LaPadula
- Problème identifié :
  - Lorsqu'on met des niveaux de sécurité dans une arborescence de fichiers
  - Est-ce que le niveau de sécurité doit être croissant ou décroissant ?

RÉPONSE :

- Décroissant : plus on s'enfonce dans l'arborescence, moins on a besoin de sécurité (comparaison / et /tmp)
- Croissant : car si on ne peut pas connaître un dossier, on ne peut rien savoir de son contenu

INTÉGRITÉ

VERSION SIMPLIFIÉE

- Bell-LaPadula permet à n'importe qui d'écrire des informations à un haut niveau de sécurité

  Carnets personnels d'un général

- Ne garantit donc pas la qualité des informations à un niveau de sécurité donné

  même exemple

- Modèle Biba : une hiérarchie permet le contrôle de la qualité d'une information

EXEMPLES

- Une rumeur est une information de très mauvaise qualité
- Un article sur Wikipédia est de bonne qualité
- Un article écrit et revu par des experts est de qualité excellente

Niveaux d'intégrité

- Une rumeur peut être basée sur un article d'experts
- Mais le contraire n'est pas possible : il faut des vérifications supplémentaires

NWU, NRD

- Comme pour Bell-LaPadula, le modèle Biba associe aux sujets et objets des niveaux d'intégrité
- Au contraire de Bell-LaPadula :
  - Un sujet peut écrire des informations à un niveau inférieur
  - Un sujet peut lire des informations qui viennent d'un niveau supérieur

Classification/déclassification dans le modèle Biba

Procédures de contrôle de qualité

- Pour faire remonter une information, il faut s'assurer que des précautions supplémentaires ont été prises au niveau inférieur
- Terme générique : procédures de contrôle de qualité, hors du champs des systèmes d'informations
- Un expert autorise le relèvement d'une information seulement si les procédures ont été bien suivies

Applications pratiques

- Intégration de COTS (*Commercial Off-The-Shelf*) dans des systèmes à haut niveau d'intégrité après *Certification*

  Critères Communs (EAL 1-7)

- Signature numérique pour le contrôle d'intégrité

Modèle logique

- Comme pour le modèle Bell-LaPadula, un treillis de niveaux d'intégrité
- règles "inversées"

Prédicats

- $\texttt{ass}(x, l)$: le sujet x a le niveau d'assurance l
- $\texttt{int}(x, l)$: l'objet x a le niveau d'intégrité l
- $\texttt{lire}(x, y)$: le sujet x peut lire l'objet y
- $\texttt{ecr}(x, y)$: le sujet x peut écrire dans l'objet y
- $l \preceq l'$: le niveau l est inférieur au niveau l'

Règles de contrôle d'accès

$$\begin{aligned} \texttt{lire}(x,y) &\leftarrow \texttt{ass}(x,l), \texttt{int}(y,l'), l \preceq l' \\ \texttt{ecr}(x,y) &\leftarrow \texttt{ass}(x,l), \texttt{int}(y,l'), l' \preceq l \\ l \preceq l' &\leftarrow l \preceq l'', l'' \preceq l' \end{aligned}$$

Modèle logique

- Comme pour le modèle Bell-LaPadula, un treillis de niveaux d'intégrité
- règles "inversées"

Prédicats

- $\texttt{ass}(x, l)$: le sujet x a le niveau d'assurance l
- $\texttt{int}(x, l)$: l'objet x a le niveau d'intégrité l
- $\texttt{lire}(x, y)$: le sujet x peut lire l'objet y
- $\texttt{ecr}(x, y)$: le sujet x peut écrire dans l'objet y
- $l \preceq l'$: le niveau l est inférieur au niveau l'

Règles de contrôle d'accès

$$\begin{aligned} \texttt{lire}(x, y) &\leftarrow \texttt{ass}(x, l), \texttt{int}(y, l'), l \preceq l' \\ \texttt{ecr}(x, y) &\leftarrow \texttt{ass}(x, l), \texttt{int}(y, l'), l' \preceq l \\ l \preceq l' &\leftarrow l \preceq l'', l'' \preceq l' \end{aligned}$$

Modèle logique

- Comme pour le modèle Bell-LaPadula, un treillis de niveaux d'intégrité
- règles "inversées"

Prédicats

- $\texttt{ass}(x, l)$: le sujet x a le niveau d'assurance l
- $\texttt{int}(x, l)$: l'objet x a le niveau d'intégrité l
- $\texttt{lire}(x, y)$: le sujet x peut lire l'objet y
- $\texttt{ecr}(x, y)$: le sujet x peut écrire dans l'objet y
- $l \preceq l'$: le niveau l est inférieur au niveau l'

Règles de contrôle d'accès

$$\begin{aligned} \texttt{lire}(x, y) &\leftarrow \texttt{ass}(x, l), \texttt{int}(y, l'), l \preceq l' \\ \texttt{ecr}(x, y) &\leftarrow \texttt{ass}(x, l), \texttt{int}(y, l'), l' \preceq l \\ l \preceq l' &\leftarrow l \preceq l'', l'' \preceq l' \end{aligned}$$

BIBA+BELL-LAPADULA

- Hypothèse : les niveaux d'intégrité et de sécurité coïncident
- No Read Up, No Read Down, No Write Up, No Write Down
- Les informations restent confinées au niveau où elles sont créées
- En pratique, contrôle d'accès pour réguler les échanges possibles d'informations
- Le mécanisme de contrôle d'accès utilisé revient à décrire les procédures de classification/déclassification

systèmes MILS (*Multiple Independent Levels of Security*)

Outline

Contrôle d'accès (théorie)

- Base logique
- Classification
- Modèle Bell-LaPadula
- Modèle Biba
- Le modèle Harrisson-Ruzzo-Ullman
- Role-Based Access Control (Généralités)

CONTEXTE

- Approches Bell-LaPadula/Biba : On décrit un système idéal, et on cherche ensuite à l'implémenter

  approche *top-down*

- Apparition des systèmes multi-utilisateurs, multi-tâches
- Approche Bottom-Up :

  on cherche à modéliser les systèmes existant

Domaine

- Système de contrôle d'accès de type Unix
- On a des sujets, qui sont tous les utilisateurs enregistrés dans le système (login et mot de passe)
- On a des objets, qui sont tous les fichiers gérés par le système
- Fichiers "Unix" : les connections réseaux, la carte son, etc. sont des fichiers

Domaine

- Système de contrôle d'accès de type Unix
- On a des sujets, qui sont tous les utilisateurs enregistrés dans le système (login et mot de passe)
- On a des objets, qui sont tous les fichiers gérés par le système
- Fichiers "Unix" : les connections réseaux, la carte son, etc. sont des fichiers

Domaine

- Système de contrôle d'accès de type Unix
- On a des sujets, qui sont tous les utilisateurs enregistrés dans le système (login et mot de passe)
- On a des objets, qui sont tous les fichiers gérés par le système
- Fichiers "Unix" : les connections réseaux, la carte son, etc. sont des fichiers

Domaine

- Système de contrôle d'accès de type Unix
- On a des sujets, qui sont tous les utilisateurs enregistrés dans le système (login et mot de passe)
- On a des objets, qui sont tous les fichiers gérés par le système
- Fichiers "Unix" : les connections réseaux, la carte son, etc. sont des fichiers

Domaine

- Système de contrôle d'accès de type Unix
- On a des sujets, qui sont tous les utilisateurs enregistrés dans le système (login et mot de passe)
- On a des objets, qui sont tous les fichiers gérés par le système
- Fichiers "Unix" : les connections réseaux, la carte son, etc. sont des fichiers

PERMISSIONS

- On considère un ensemble $a_1, \ldots, a_n$ d'actions
- On associe une permission à chaque action
- Une permission, c'est une relation entre les sujets et les objets
- On a une représentation matricielle

PERMISSIONS

- On considère un ensemble $a_1, \ldots, a_n$ d'actions
- On associe une permission à chaque action
- Une permission, c'est une relation entre les sujets et les objets
- On a une représentation matricielle

PERMISSIONS

- On considère un ensemble $a_1, \ldots, a_n$ d'actions
- On associe une permission à chaque action
- Une permission, c'est une relation entre les sujets et les objets
- On a une représentation matricielle

PERMISSIONS

- On considère un ensemble $a_1, \ldots, a_n$ d'actions
- On associe une permission à chaque action
- Une permission, c'est une relation entre les sujets et les objets
- On a une représentation matricielle

PERMISSIONS

- On considère un ensemble $a_1, \ldots, a_n$ d'actions
- On associe une permission à chaque action
- Une permission, c'est une relation entre les sujets et les objets
- On a une représentation matricielle

Représentation matricielle

sujet| objet	o_1	...	o_n
s_1	a_1	...	a_2
$\vdots$	$\vdots$	...	$\vdots$
s_k	a_1, a_2	...	a_1, a_3

$a_1, \ldots, a_n$ sont des actions

COMMANDES

CONTRÔLE D'ACCÈS À DISCRÉTION

- une action spéciale représente le fait de "posséder" un objet
- Des *commandes* indiquent les modifications possibles dans la matrice de contrôle d'accès
- Si ces commandes peuvent aussi ajouter des objets et des fichiers, savoir si une permission peut être obtenue devient indécidable

COMMANDES

CONTRÔLE D'ACCÈS À DISCRÉTION

- une action spéciale représente le fait de "posséder" un objet
- Des *commandes* indiquent les modifications possibles dans la matrice de contrôle d'accès
- Si ces commandes peuvent aussi ajouter des objets et des fichiers, savoir si une permission peut être obtenue devient indécidable

COMMANDES

CONTRÔLE D'ACCÈS À DISCRÉTION

- une action spéciale représente le fait de "posséder" un objet
- Des *commandes* indiquent les modifications possibles dans la matrice de contrôle d'accès
- Si ces commandes peuvent aussi ajouter des objets et des fichiers, savoir si une permission peut être obtenue devient indécidable

COMMANDES

CONTRÔLE D'ACCÈS À DISCRÉTION

- une action spéciale représente le fait de "posséder" un objet
- Des *commandes* indiquent les modifications possibles dans la matrice de contrôle d'accès
- Si ces commandes peuvent aussi ajouter des objets et des fichiers, savoir si une permission peut être obtenue devient indécidable

COMMANDES

COMMANDES ATOMIQUES

- **enter** r **into** $M(s,o)$
- **delete** r **into** $M(s,o)$
- **create subject** s
- **delete subject** s
- **create object** o
- **delete object** o

COMMANDES

- **if** r **in** $M(s,o)$ **then**
- + séquence de commandes atomiques

Exemple de commande HRU

Création de fichier

Command create_file(s, f)

create object f

enter o **into** $M(s, f)$

enter r **into** $M(s, f)$

enter w **into** $M(s, f)$

EXEMPLE DE COMMANDE HRU

PASSATION DU DROIT DE LECTURE

Command grant_read(s, s', f)

if $o \in M(s, f)$ **then**

enter r **into** $M(s', f)$

Évaluation

Problème

- On se donne :
  - une matrice de contrôle d'accès initial M_0
  - un ensemble de commandes
  - un droit r
- Et on cherche à savoir s'il existe un couple (s, o) et une séquence de commandes telles qu'après ces commandes, $M(s, o)$ contient r alors que $r \notin M_0(s, o)$

Résultats

- Ce problème est décidable si chaque commande ne contient qu'une seule action
- Il est indécidable dans le cas général
- Il est indécidable même si toutes les actions ne font qu'ajouter des droits (jolie preuve)

Modèle logique

Prédicats :

- Pour chaque droit r un prédicat $r(x, y)$, avec $r(s, o)$ vrai si $r \in M(s, o)$
- Pour chaque commande $c(x_1, \ldots, x_n)$, un prédicat $\texttt{permis\_c}(x_1, \ldots, x_n)$

Permissions pour les commandes

Pour chaque commande, une clause de Horn :

- Traduction des conditions $r \in M(s, o)$ par un atome dans le corps de la clause
- $\texttt{permis\_c}(x_1, \ldots, x_n)$ dans la tête de la clause

Exemple

$$\texttt{permis\_grant\_read}(s, s', f) \leftarrow o(s, f)$$

Modèle dynamique

Effet des commandes

- Chaque commande modifie la matrice de contrôle d'accès
- Un *état* est défini par une matrice de contrôle d'accès
- Les commandes font passer d'un état à un autre
- L'évaluation des permissions peut changer lorsqu'on change d'état

Définition de l'état suivant

- $\texttt{permis\_c}(x_1, \ldots, x_n) \Rightarrow (E, A)$
- E est un ensemble d'atomes à enlever
- A est un ensemble d'atomes à ajouter
- Hypothèse : Pour chaque commande :

  remove r **from** (s, o)

  il y a un test $r \in (s, o)$

COMMENTAIRES

- D'autres modèles avec des commandes plus simples existent
- En pratique, ce sont des modèles "anciens", car les droits doivent être exprimés pour tous les utilisateurs
- On préfère utiliser des systèmes plus modernes
- Pour les systèmes à sécuriser, il est aussi préférable que les utilisateurs ne puissent pas changer les droits d'accès
- Histoire du contrôle d'accès : autoriser des changements tout en ayant une évaluation décidable

POINT IMPORTANT

- Autoriser les modifications de l'état courant permet de tout coder (problème indécidable)
- Il faut encadrer les modifications possibles

COMMENTAIRES

- D'autres modèles avec des commandes plus simples existent
- En pratique, ce sont des modèles "anciens", car les droits doivent être exprimés pour tous les utilisateurs
- On préfère utiliser des systèmes plus modernes
- Pour les systèmes à sécuriser, il est aussi préférable que les utilisateurs ne puissent pas changer les droits d'accès
- Histoire du contrôle d'accès : autoriser des changements tout en ayant une évaluation décidable

POINT IMPORTANT

- Autoriser les modifications de l'état courant permet de tout coder (problème indécidable)
- Il faut encadrer les modifications possibles

COMMENTAIRES

- D'autres modèles avec des commandes plus simples existent
- En pratique, ce sont des modèles "anciens", car les droits doivent être exprimés pour tous les utilisateurs
- On préfère utiliser des systèmes plus modernes
- Pour les systèmes à sécuriser, il est aussi préférable que les utilisateurs ne puissent pas changer les droits d'accès
- Histoire du contrôle d'accès : autoriser des changements tout en ayant une évaluation décidable

POINT IMPORTANT

- Autoriser les modifications de l'état courant permet de tout coder (problème indécidable)
- Il faut encadrer les modifications possibles

COMMENTAIRES

- D'autres modèles avec des commandes plus simples existent
- En pratique, ce sont des modèles "anciens", car les droits doivent être exprimés pour tous les utilisateurs
- On préfère utiliser des systèmes plus modernes
- Pour les systèmes à sécuriser, il est aussi préférable que les utilisateurs ne puissent pas changer les droits d'accès
- Histoire du contrôle d'accès : autoriser des changements tout en ayant une évaluation décidable

POINT IMPORTANT

- Autoriser les modifications de l'état courant permet de tout coder (problème indécidable)
- Il faut encadrer les modifications possibles

COMMENTAIRES

- D'autres modèles avec des commandes plus simples existent
- En pratique, ce sont des modèles "anciens", car les droits doivent être exprimés pour tous les utilisateurs
- On préfère utiliser des systèmes plus modernes
- Pour les systèmes à sécuriser, il est aussi préférable que les utilisateurs ne puissent pas changer les droits d'accès
- Histoire du contrôle d'accès : autoriser des changements tout en ayant une évaluation décidable

POINT IMPORTANT

- Autoriser les modifications de l'état courant permet de tout coder (problème indécidable)
- Il faut encadrer les modifications possibles

COMMENTAIRES

- D'autres modèles avec des commandes plus simples existent
- En pratique, ce sont des modèles "anciens", car les droits doivent être exprimés pour tous les utilisateurs
- On préfère utiliser des systèmes plus modernes
- Pour les systèmes à sécuriser, il est aussi préférable que les utilisateurs ne puissent pas changer les droits d'accès
- Histoire du contrôle d'accès : autoriser des changements tout en ayant une évaluation décidable

POINT IMPORTANT

- Autoriser les modifications de l'état courant permet de tout coder (problème indécidable)
- Il faut encadrer les modifications possibles

Outline

Contrôle d'accès (théorie)

GROUPES ET RÔLES

NOTION DE GROUPE

- un *groupe* est un ensemble d'utilisateurs
- Associer des droits à un groupe permet de ne pas avoir à le faire pour tous les utilisateurs du groupe

NOTION DE RÔLE

- Les rôles sont des groupes
- différence : hiérarchie de rôles
  - Comme pour les objets, un rôle peut *hériter* des permissions d'un autre rôle
  - intérêt 1 : refléter, dans la politique de contrôle d'accès, la hiérarchie au sein d'une organisation
  - intérêt 2 : définir des rôles par rapport à des activités

GROUPES ET RÔLES

NOTION DE GROUPE

- un *groupe* est un ensemble d'utilisateurs
- Associer des droits à un groupe permet de ne pas avoir à le faire pour tous les utilisateurs du groupe

NOTION DE RÔLE

- Les rôles sont des groupes
- différence : hiérarchie de rôles
  - Comme pour les objets, un rôle peut *hériter* des permissions d'un autre rôle
  - intérêt 1 : refléter, dans la politique de contrôle d'accès, la hiérarchie au sein d'une organisation
  - intérêt 2 : définir des rôles par rapport à des activités

GROUPES ET RÔLES

NOTION DE GROUPE

- un *groupe* est un ensemble d'utilisateurs
- Associer des droits à un groupe permet de ne pas avoir à le faire pour tous les utilisateurs du groupe

NOTION DE RÔLE

- Les rôles sont des groupes
- différence : hiérarchie de rôles
  - Comme pour les objets, un rôle peut *hériter* des permissions d'un autre rôle
  - intérêt 1 : refléter, dans la politique de contrôle d'accès, la hiérarchie au sein d'une organisation
  - intérêt 2 : définir des rôles par rapport à des activités

GROUPES ET RÔLES

NOTION DE GROUPE

- un *groupe* est un ensemble d'utilisateurs
- Associer des droits à un groupe permet de ne pas avoir à le faire pour tous les utilisateurs du groupe

NOTION DE RÔLE

- Les rôles sont des groupes
- différence : hiérarchie de rôles
  - Comme pour les objets, un rôle peut *hériter* des permissions d'un autre rôle
  - intérêt 1 : refléter, dans la politique de contrôle d'accès, la hiérarchie au sein d'une organisation
  - intérêt 2 : définir des rôles par rapport à des activités

Role-Based Access Control (RBAC) (1/2)

Briques de base d'un langage pour ,le contrôle d'accès

- utilisateurs : les personnes ayant accès au système
- sujets : les programmes personnifiant l'utilisateur au sein du système
- actions : opérations définies dans le système
- rôles : ensemble de noms avec une hiérarchie $\prec$. Si $R_1 \prec r_2$, alors le rôle r_2 hérite des permissions du rôle r_1

Role-Based Access Control (RBAC) (1/2)

Briques de base d'un langage pour ,le contrôle d'accès

- utilisateurs : les personnes ayant accès au système
- sujets : les programmes personnifiant l'utilisateur au sein du système
- actions : opérations définies dans le système
- rôles : ensemble de noms avec une hiérarchie $\prec$. Si $R_1 \prec r_2$, alors le rôle r_2 hérite des permissions du rôle r_1

Role-Based Access Control (RBAC) (1/2)

Briques de base d'un langage pour ,le contrôle d'accès

- utilisateurs : les personnes ayant accès au système
- sujets : les programmes personnifiant l'utilisateur au sein du système
- actions : opérations définies dans le système
- rôles : ensemble de noms avec une hiérarchie $\prec$. Si $R_1 \prec r_2$, alors le rôle r_2 hérite des permissions du rôle r_1

Role-Based Access Control (RBAC) (1/2)

Briques de base d'un langage pour ,le contrôle d'accès

- utilisateurs : les personnes ayant accès au système
- sujets : les programmes personnifiant l'utilisateur au sein du système
- actions : opérations définies dans le système
- rôles : ensemble de noms avec une hiérarchie $\prec$. Si $R_1 \prec r_2$, alors le rôle r_2 hérite des permissions du rôle r_1

Role-Based Access Control (RBAC) (1/2)

Briques de base d'un langage pour ,le contrôle d'accès

- utilisateurs : les personnes ayant accès au système
- sujets : les programmes personnifiant l'utilisateur au sein du système
- actions : opérations définies dans le système
- rôles : ensemble de noms avec une hiérarchie $\prec$. Si $R_1 \prec r_2$, alors le rôle r_2 hérite des permissions du rôle r_1

Role-Based Access Control (RBAC) (2/2)

Relations définies dans RBAC

- session : sujets utilisables par un utilisateur (relation $1 - n$)
- permissions : relation $n - n$ entre les rôles et les actions indiquant les actions permises pour chaque rôle. En général (et dans la suite) les actions sont effectués sur des objets
- peut-jouer : relation $n - n$ entre les sujets et les rôles indiquant quels rôles (*i.e.* ensemble de permissions) sont accessibles à un sujet
- joue : relation $n - n$ indiquant les rôles utilisés par un sujet (un sujet doit jouer un rôle pour pouvoir utiliser les permissions de ce rôle)

Role-Based Access Control (RBAC) (2/2)

Relations définies dans RBAC

- session : sujets utilisables par un utilisateur (relation $1 - n$)
- permissions : relation $n - n$ entre les rôles et les actions indiquant les actions permises pour chaque rôle. En général (et dans la suite) les actions sont effectués sur des objets
- peut-jouer : relation $n - n$ entre les sujets et les rôles indiquant quels rôles (*i.e.* ensemble de permissions) sont accessibles à un sujet
- joue : relation $n - n$ indiquant les rôles utilisés par un sujet (un sujet doit jouer un rôle pour pouvoir utiliser les permissions de ce rôle)

Role-Based Access Control (RBAC) (2/2)

Relations définies dans RBAC

- session : sujets utilisables par un utilisateur (relation $1 - n$)
- permissions : relation $n - n$ entre les rôles et les actions indiquant les actions permises pour chaque rôle. En général (et dans la suite) les actions sont effectués sur des objets
- peut-jouer : relation $n - n$ entre les sujets et les rôles indiquant quels rôles (*i.e.* ensemble de permissions) sont accessibles à un sujet
- joue : relation $n - n$ indiquant les rôles utilisés par un sujet (un sujet doit jouer un rôle pour pouvoir utiliser les permissions de ce rôle)

Role-Based Access Control (RBAC) (2/2)

Relations définies dans RBAC

- session : sujets utilisables par un utilisateur (relation $1 - n$)
- permissions : relation $n - n$ entre les rôles et les actions indiquant les actions permises pour chaque rôle. En général (et dans la suite) les actions sont effectués sur des objets
- peut-jouer : relation $n - n$ entre les sujets et les rôles indiquant quels rôles (*i.e.* ensemble de permissions) sont accessibles à un sujet
- joue : relation $n - n$ indiquant les rôles utilisés par un sujet (un sujet doit jouer un rôle pour pouvoir utiliser les permissions de ce rôle)

Role-Based Access Control (RBAC) (2/2)

Relations définies dans RBAC

- session : sujets utilisables par un utilisateur (relation $1 - n$)
- permissions : relation $n - n$ entre les rôles et les actions indiquant les actions permises pour chaque rôle. En général (et dans la suite) les actions sont effectués sur des objets
- peut-jouer : relation $n - n$ entre les sujets et les rôles indiquant quels rôles (*i.e.* ensemble de permissions) sont accessibles à un sujet
- joue : relation $n - n$ indiquant les rôles utilisés par un sujet (un sujet doit jouer un rôle pour pouvoir utiliser les permissions de ce rôle)

Quelques remarques sur RBAC (1/2)

Remarques techniques

- Linux : newgrp permet de jouer un nouveau groupe. L'activation d'un nouveau groupe peut être contrôlée par un mot de passe.
- Windows : il y a des *domaines* et des *groupes locaux*. Un domaine peut être membre d'un groupe local, ce qui permet d'exprimer un héritage de permission dans un sens
- ACL : suivant l'implémentation, un groupe peut être membre d'un groupe. Là encore, on peut exprimer l'héritage de permissions

Quelques remarques sur RBAC (1/2)

Remarques techniques

- Linux : newgrp permet de jouer un nouveau groupe. L'activation d'un nouveau groupe peut être contrôlée par un mot de passe.
- Windows : il y a des *domaines* et des *groupes locaux*. Un domaine peut être membre d'un groupe local, ce qui permet d'exprimer un héritage de permission dans un sens
- ACL : suivant l'implémentation, un groupe peut être membre d'un groupe. Là encore, on peut exprimer l'héritage de permissions

Quelques remarques sur RBAC (1/2)

Remarques techniques

- Linux : newgrp permet de jouer un nouveau groupe. L'activation d'un nouveau groupe peut être contrôlée par un mot de passe.
- Windows : il y a des *domaines* et des *groupes locaux*. Un domaine peut être membre d'un groupe local, ce qui permet d'exprimer un héritage de permission dans un sens
- ACL : suivant l'implémentation, un groupe peut être membre d'un groupe. Là encore, on peut exprimer l'héritage de permissions

Quelques remarques sur RBAC (1/2)

Remarques techniques

- Linux : newgrp permet de jouer un nouveau groupe. L'activation d'un nouveau groupe peut être contrôlée par un mot de passe.
- Windows : il y a des *domaines* et des *groupes locaux*. Un domaine peut être membre d'un groupe local, ce qui permet d'exprimer un héritage de permission dans un sens
- ACL : suivant l'implémentation, un groupe peut être membre d'un groupe. Là encore, on peut exprimer l'héritage de permissions

Quelques remarques sur RBAC (2/2)

Historique

- RBAC d'abord implémenté pour l'accès aux bases de données (fin des années 80)
- Formalisé, puis standardisé par le NIST, dans les années 90
- Utilisé dans tous les systèmes modernes

Utilisateurs et sujets

Souvent, on ne fait pas de différences entre utilisateur et sujet au niveau de la politique de contrôle d'accès. La distinction est utile quand on veut spécifier comment le lien peut être fait (par exemple, utilisation d'un mot de passe et *via* une communication locale)

Quelques remarques sur RBAC (2/2)

Historique

- RBAC d'abord implémenté pour l'accès aux bases de données (fin des années 80)
- Formalisé, puis standardisé par le NIST, dans les années 90
- Utilisé dans tous les systèmes modernes

Utilisateurs et sujets

Souvent, on ne fait pas de différences entre utilisateur et sujet au niveau de la politique de contrôle d'accès. La distinction est utile quand on veut spécifier comment le lien peut être fait (par exemple, utilisation d'un mot de passe et *via* une communication locale)

EXERCICE : POLITIQUE DANS UN HÔPITAL

DÉFINITION DES RÔLES

- Rôles «médicaux» : professeur, médecin, interne, infirmier
- Rôles administratif : comptable, directeur, réceptionniste
- Autres : patient, visiteur

Question 1 : Donner une hiérarchie sur ces rôles

EXERCICE : POLITIQUE DANS UN HÔPITAL

- Cyrus est professeur, Hiram médecin et Mieszko infirmier
- Afonso est le directeur, Champlain est comptable et Caesar réceptionniste

- Catherine et Henry ne sont rien de tout ça

Question 2 : Donner la relation utilisateurs/rôles

EXERCICE : POLITIQUE DANS UN HÔPITAL

- Cyrus est professeur, Hiram médecin et Mieszko infirmier
- Afonso est le directeur, Champlain est comptable et Caesar réceptionniste

- Catherine et Henry ne sont rien de tout ça

Question 2 : Donner la relation utilisateurs/rôles

EXERCICE : POLITIQUE DANS UN HÔPITAL

- Cyrus est professeur, Hiram médecin et Mieszko infirmier
- Afonso est le directeur, Champlain est comptable et Caesar réceptionniste

- Catherine et Henry ne sont rien de tout ça

Question 2 : Donner la relation utilisateurs/rôles

EXERCICE : POLITIQUE DANS UN HÔPITAL

- Cyrus est professeur, Hiram médecin et Mieszko infirmier
- Afonso est le directeur, Champlain est comptable et Caesar réceptionniste

- Catherine et Henry ne sont rien de tout ça

Question 2 : Donner la relation utilisateurs/rôles

SÉPARATION DES TÂCHES

Dans l'exemple précédent, Cyrus peut être á la fois professeur et patient et visiteur. En pratique :

- on peut vouloir que personne ne puisse jouer deux rôles spécifiques
  - exemple intuitif : un joueur ne peut pas être en même au Stade Français et au Stade Toulousain
  - exemple réaliste : Sarbanes-Oxley/muraille de Chine/...
- On peut vouloir qu'il ne soit pas possible de jouer deux rôles en même temps
  - Un médecin ne peut pas être aussi patient
  - ...

On parle de *Séparation des tâches* statique (SSoD) ou dynamique (DSoD)

Question 3 : Quels rôles devraient être en séparation statique ou dynamique des tâches

SÉPARATION DES TÂCHES

Dans l'exemple précédent, Cyrus peut être á la fois professeur et patient et visiteur. En pratique :

- on peut vouloir que personne ne puisse jouer deux rôles spécifiques
  - exemple intuitif : un joueur ne peut pas être en même au Stade Français et au Stade Toulousain
  - exemple réaliste : Sarbanes-Oxley/muraille de Chine/...
- On peut vouloir qu'il ne soit pas possible de jouer deux rôles en même temps
  - Un médecin ne peut pas être aussi patient
  - ...

On parle de *Séparation des tâches* statique (SSoD) ou dynamique (DSoD)

Question 3 : Quels rôles devraient être en séparation statique ou dynamique des tâches

SÉPARATION DES TÂCHES

Dans l'exemple précédent, Cyrus peut être á la fois professeur et patient et visiteur. En pratique :

- on peut vouloir que personne ne puisse jouer deux rôles spécifiques
  - exemple intuitif : un joueur ne peut pas être en même au Stade Français et au Stade Toulousain
  - exemple réaliste : Sarbanes-Oxley/muraille de Chine/...
- On peut vouloir qu'il ne soit pas possible de jouer deux rôles en même temps
  - Un médecin ne peut pas être aussi patient
  - ...

On parle de *Séparation des tâches* statique (SSoD) ou dynamique (DSoD)

Question 3 : Quels rôles devraient être en séparation statique ou dynamique des tâches

Théorême de séparation

Theorem

Deux rôles en séparation statique ne peuvent hériter des permissions l'un de l'autre

Preuve

- Un rôle est un ensemble de permissions
- puis définition de séparation statique.

Théorême de séparation

Theorem

Deux rôles en séparation statique ne peuvent hériter des permissions l'un de l'autre

Preuve

- Un rôle est un ensemble de permissions
- puis définition de séparation statique.

Morale du théorème de séparation

- Il faut faire attention, la séparation de tâches se marie mal avec l'héritage
- ex : conflits patient/médecin, visiteur/patient,. . .
- Solution 1 (Solaris, SELinux) : strict contrôle sur la hiérarchie des rôles (aucune, ou un seul étage)
- Solution 2 : Ne pas définir la hiérarchie de rôles par rapport aux fonctions (médecin, etc.) mais par rapport aux activités.
- Formalisation (ou non) de ces activités avec un *environnement* qui désigne la tâche en cours.

Morale du théorème de séparation

- Il faut faire attention, la séparation de tâches se marie mal avec l'héritage
- ex : conflits patient/médecin, visiteur/patient,...
- Solution 1 (Solaris, SELinux) : strict contrôle sur la hiérarchie des rôles (aucune, ou un seul étage)
- Solution 2 : Ne pas définir la hiérarchie de rôles par rapport aux fonctions (médecin, etc.) mais par rapport aux activités.
- Formalisation (ou non) de ces activités avec un *environnement* qui désigne la tâche en cours.

Morale du théorème de séparation

- Il faut faire attention, la séparation de tâches se marie mal avec l'héritage
- ex : conflits patient/médecin, visiteur/patient,...
- Solution 1 (Solaris, SELinux) : strict contrôle sur la hiérarchie des rôles (aucune, ou un seul étage)
- Solution 2 : Ne pas définir la hiérarchie de rôles par rapport aux fonctions (médecin, etc.) mais par rapport aux activités.
- Formalisation (ou non) de ces activités avec un *environnement* qui désigne la tâche en cours.

Morale du théorème de séparation

- Il faut faire attention, la séparation de tâches se marie mal avec l'héritage
- ex : conflits patient/médecin, visiteur/patient,...
- Solution 1 (Solaris, SELinux) : strict contrôle sur la hiérarchie des rôles (aucune, ou un seul étage)
- Solution 2 : Ne pas définir la hiérarchie de rôles par rapport aux fonctions (médecin, etc.) mais par rapport aux activités.
- Formalisation (ou non) de ces activités avec un *environnement* qui désigne la tâche en cours.

Morale du théorème de séparation

- Il faut faire attention, la séparation de tâches se marie mal avec l'héritage
- ex : conflits patient/médecin, visiteur/patient,...
- Solution 1 (Solaris, SELinux) : strict contrôle sur la hiérarchie des rôles (aucune, ou un seul étage)
- Solution 2 : Ne pas définir la hiérarchie de rôles par rapport aux fonctions (médecin, etc.) mais par rapport aux activités.
- Formalisation (ou non) de ces activités avec un *environnement* qui désigne la tâche en cours.

Morale du théorème de séparation

- Il faut faire attention, la séparation de tâches se marie mal avec l'héritage
- ex : conflits patient/médecin, visiteur/patient,...
- Solution 1 (Solaris, SELinux) : strict contrôle sur la hiérarchie des rôles (aucune, ou un seul étage)
- Solution 2 : Ne pas définir la hiérarchie de rôles par rapport aux fonctions (médecin, etc.) mais par rapport aux activités.
- Formalisation (ou non) de ces activités avec un *environnement* qui désigne la tâche en cours.

PERMISSIONS

RAPPEL

- Relation Perm($Role, Action, Objet$)
- En pratique, les objets eux-mêmes sont souvent des fonctions. . .
- Le système doit garder la trace des rôles activés par un sujet s. C'est représenté par le prédicat joue(s, r)
- Règle générale pour savoir si un sujet s peut faire une action a sur un objet o :

$$\text{perm}(s, a, o) \leftarrow \text{joue}(s, r), \text{Perm}(r, a, o)$$

Activation/désactivation (1/2)

Focus sur le rôle visiteur

- La réponse à l'exercice d'avant était que tout le monde peut être visiteur dans l'hôpital
- Dans le détail, il y a quand même des contraintes de temps, par exemple il doit être entre 14h et 17h
- Donc tout le monde *peut* jouer le rôle visiteur, mais pour l'activer, il faudra faire un test supplémentaire
- *activer* un rôle est aussi une action, et elle est aussi soumise au contrôle d'accès (récent)
- On est en train de rentrer dans les «détails d'implémentation», qui varient suivant le système effectivement utilisé

Activation/désactivation (2/2)

Example

Exemple

$$
\begin{aligned}
\text{joue}(s,r) &\leftarrow \text{peut-jouer}(s,r),\\
&\quad\ 14h \leq \textit{Heure} \leq 17h\\
&\text{ou}\\
\text{joue}(s,r) &\leftarrow \text{perm}(s,\text{active},r)\\
\text{Perm}(x,\text{active},\text{patient}) &\leftarrow 14h \leq \textit{Heure} \leq 17h
\end{aligned}
$$

Écrire des politiques de contrôle d'accès

- En reprenant l'exemple du patient, il y a peut-être des contrôles à faire à la réception
- Avant d'écrire une politique, il faut commencer par identifier les processus métiers (*Business Process*), et voir comment assurer que ces BP sont correctement exécutés
- En général des outils existent pour tester la politique qu'on écrit
- Dans la suite,
  - Quelques exercices : comment spécifier une politique RBAC avec des processus métiers
  - politiques de contrôle d'accès pour l'informatique dans le cas de services Web protégés par XACML

Écrire des politiques de contrôle d'accès

- En reprenant l'exemple du patient, il y a peut-être des contrôles à faire à la réception
- Avant d'écrire une politique, il faut commencer par identifier les processus métiers (*Business Process*), et voir comment assurer que ces BP sont correctement exécutés
- En général des outils existent pour tester la politique qu'on écrit
- Dans la suite,
  - Quelques exercices : comment spécifier une politique RBAC avec des processus métiers
  - politiques de contrôle d'accès pour l'informatique dans le cas de services Web protégés par XACML

Écrire des politiques de contrôle d'accès

- En reprenant l'exemple du patient, il y a peut-être des contrôles à faire à la réception
- Avant d'écrire une politique, il faut commencer par identifier les processus métiers (*Business Process*), et voir comment assurer que ces BP sont correctement exécutés
- En général des outils existent pour tester la politique qu'on écrit
- Dans la suite,
  - Quelques exercices : comment spécifier une politique RBAC avec des processus métiers
  - politiques de contrôle d'accès pour l'informatique dans le cas de services Web protégés par XACML

Écrire des politiques de contrôle d'accès

- En reprenant l'exemple du patient, il y a peut-être des contrôles à faire à la réception
- Avant d'écrire une politique, il faut commencer par identifier les processus métiers (*Business Process*), et voir comment assurer que ces BP sont correctement exécutés
- En général des outils existent pour tester la politique qu'on écrit
- Dans la suite,
  - Quelques exercices : comment spécifier une politique RBAC avec des processus métiers
  - politiques de contrôle d'accès pour l'informatique dans le cas de services Web protégés par XACML

Écrire des politiques de contrôle d'accès

- En reprenant l'exemple du patient, il y a peut-être des contrôles à faire à la réception
- Avant d'écrire une politique, il faut commencer par identifier les processus métiers (*Business Process*), et voir comment assurer que ces BP sont correctement exécutés
- En général des outils existent pour tester la politique qu'on écrit
- Dans la suite,
  - Quelques exercices : comment spécifier une politique RBAC avec des processus métiers
  - politiques de contrôle d'accès pour l'informatique dans le cas de services Web protégés par XACML

Plan

Outline

Applications Web

Modèle MVC

Modèle

- Une application Web est organisée autour d'une base de donnée
- Ce sont ces données qui sont la spécificité d'un site (github)

Vue

- Les réponses aux requêtes sont des messages ou des pages Web
- Le contenu de ces pages doit refléter les permissions du client

Contrôle

- Le routage des requêtes vers les pages et les actions internes
- Instanciation des paramêtres utilisés dans les vues

Modèle MVC

Modèle

- Une application Web est organisée autour d'une base de donnée
- Ce sont ces données qui sont la spécificité d'un site (github)

Vue

- Les réponses aux requêtes sont des messages ou des pages Web
- Le contenu de ces pages doit refléter les permissions du client

Contrôle

- Le routage des requêtes vers les pages et les actions internes
- Instanciation des paramêtres utilisés dans les vues

Modèle MVC

Modèle

- Une application Web est organisée autour d'une base de donnée
- Ce sont ces données qui sont la spécificité d'un site (github)

Vue

- Les réponses aux requêtes sont des messages ou des pages Web
- Le contenu de ces pages doit refléter les permissions du client

Contrôle

- Le routage des requêtes vers les pages et les actions internes
- Instanciation des paramêtres utilisés dans les vues

CONTRÔLE D'ACCÈS DANS LE MODÈLE MVC

PRINCIPE

- Les règles de contrôle d'accès sont édictées au niveau des données
- Ces règles sont prises en compte dans les vues pour l'affichage ou non de certaines portions
- Il est de mauvais goût de placer des règles de contrôle d'accès dans les contrôleurs

Suite : contrôle d'accès avec Rails+Hobo (pour débutants)

CONTRÔLE D'ACCÈS DANS LE MODÈLE MVC

PRINCIPE

- Les règles de contrôle d'accès sont édictées au niveau des données
- Ces règles sont prises en compte dans les vues pour l'affichage ou non de certaines portions
- Il est de mauvais goût de placer des règles de contrôle d'accès dans les contrôleurs

Suite : contrôle d'accès avec Rails+Hobo (pour débutants)

Ressources

Une ressource est :

- Un objet (dont on spécifie les attributs et leur type)
- Une table (dont les colonnes sont les attributs + admin)
- Un contrôleur (actions index, new, create, show, edit, update, destroy)
- Une vue par réponse possible (*e.g.*, la vue correspondant à `new` ou à `edit` est un formulaire, celles de `create` et `update` sont par défaut des redirections vers la page `show`)

Hobo

- Une intégration des règles de contrôle d'accès dans les modèles
- Une librairie pour l'affichage des données qui prend en compte le contrôle d'accès sur les modèles

CONTRÔLE D'ACCÈS : MODÈLE

LE SUJET EST DÉNOMMÉ ACTING_USER

```
class User < ActiveRecord::Base

  hobo_user_model # Don't put anything above this

  "..."
  def create_permitted?
    # Only the initial
    # admin user can be created
    self.class.count == 0
  end

  def update_permitted?
    true
    || ( acting_user.administrator?
    || acting_user.has_role("Animateur") ) && "..."
  end

  def destroy_permitted?
    acting_user.administrator?
  end

  def view_permitted?(field)
   true || (acting_user == self) ||
   (acting_user.administrator?) ||
   (field == :name ) || (field == :role_id )
  end
end
```

COMMENTAIRES

VUES ET CONTRÔLE D'ACCÈS

- Les opérations possibles ne sont pas impactées
- Les opérations non-réalisables sont justes cachées
- But : ne pas distraire l'utilisateur avec des boutons qu'il n'a pas le droit d'utiliser
- Sécurité : pas de changement qu'on présente ou non les boutons

TP OPTIONNEL À FAIRE CHEZ SOI

- Créer les ressources pour RBAC (quelles tables ?)
- Écrire les règles génériques pour le contrôle d'accès

COMMENTAIRES

VUES ET CONTRÔLE D'ACCÈS

- Les opérations possibles ne sont pas impactées
- Les opérations non-réalisables sont justes cachées
- But : ne pas distraire l'utilisateur avec des boutons qu'il n'a pas le droit d'utiliser
- Sécurité : pas de changement qu'on présente ou non les boutons

TP OPTIONNEL À FAIRE CHEZ SOI

- Créer les ressources pour RBAC (quelles tables ?)
- Écrire les règles génériques pour le contrôle d'accès

Outline

Applications Web

Processus métiers

C'est une activité :

- Abréviation : BP (*Business Process*)
- Définie par un certain nombre de tâches à accomplir
- Un ordre sur ces tâches
- Des conditions à vérifier
- ...

Modélisation habituelle :

diagramme de tâches (*workflow*)

Processus métiers

C'est une activité :

- Abréviation : BP (*Business Process*)
- Définie par un certain nombre de tâches à accomplir
- Un ordre sur ces tâches
- Des conditions à vérifier
- ...

Modélisation habituelle :

diagramme de tâches (*workflow*)

Processus métiers

C'est une activité :

- Abréviation : BP (*Business Process*)
- Définie par un certain nombre de tâches à accomplir
- Un ordre sur ces tâches
- Des conditions à vérifier
- ...

Modélisation habituelle :

diagramme de tâches (*workflow*)

Processus métiers

C'est une activité :

- Abréviation : BP (*Business Process*)
- Définie par un certain nombre de tâches à accomplir
- Un ordre sur ces tâches
- Des conditions à vérifier
- ...

Modélisation habituelle :

diagramme de tâches (*workflow*)

Processus métiers

C'est une activité :

- Abréviation : BP (*Business Process*)
- Définie par un certain nombre de tâches à accomplir
- Un ordre sur ces tâches
- Des conditions à vérifier
- ...

Modélisation habituelle :

diagramme de tâches (*workflow*)

Processus métiers

C'est une activité :

- Abréviation : BP (*Business Process*)
- Définie par un certain nombre de tâches à accomplir
- Un ordre sur ces tâches
- Des conditions à vérifier
- ...

Modélisation habituelle :

diagramme de tâches (*workflow*)

CONTRÔLE D'ACCÈS DANS LES PROCESSUS MÉTIERS (1/2)

PRINCIPE

- Chaque BP doit définir un certain nombre de rôles
- Les tâches sont réparties entre ces rôles
- Il faut vérifier que les personnes qui exécutent les tâches peuvent effectivement être actives dans les rôles qui sont définis

WORKFLOW ET CONTRÔLE D'ACCÈS

- un diagramme d'exécution, avec une distribution des tâches aux différents rôles
- Un système de contrôle d'accès contrôlant l'instanciation des rôles par des sujets
- Ce dernier doit faire le pont entre des rôles organisationnels (réceptionniste), et des rôles au sein du BP

CONTRÔLE D'ACCÈS DANS LES PROCESSUS MÉTIERS (1/2)

PRINCIPE

- Chaque BP doit définir un certain nombre de rôles
- Les tâches sont réparties entre ces rôles
- Il faut vérifier que les personnes qui exécutent les tâches peuvent effectivement être actives dans les rôles qui sont définis

WORKFLOW ET CONTRÔLE D'ACCÈS

- un diagramme d'exécution, avec une distribution des tâches aux différents rôles
- Un système de contrôle d'accès contrôlant l'instanciation des rôles par les sujets
- Ce dernier doit faire le pont entre des rôles organisationnels (réception mail), et des rôles au sein du BP

CONTRÔLE D'ACCÈS DANS LES PROCESSUS MÉTIERS (1/2)

PRINCIPE

- Chaque BP doit définir un certain nombre de rôles
- Les tâches sont réparties entre ces rôles
- Il faut vérifier que les personnes qui exécutent les tâches peuvent effectivement être actives dans les rôles qui sont définis

WORKFLOW ET CONTRÔLE D'ACCÈS

- un diagramme d'exécution, avec une distribution des tâches aux différents rôles
- Un système de contrôle d'accès contrôlant l'instanciation des rôles par des sujets
- Ce dernier doit faire le pont entre des rôles organisationnels (réceptionniste), et des rôles au sein du BP

CONTRÔLE D'ACCÈS DANS LES PROCESSUS MÉTIERS (1/2)

PRINCIPE

- Chaque BP doit définir un certain nombre de rôles
- Les tâches sont réparties entre ces rôles
- Il faut vérifier que les personnes qui exécutent les tâches peuvent effectivement être actives dans les rôles qui sont définis

WORKFLOW ET CONTRÔLE D'ACCÈS

- un diagramme d'exécution, avec une distribution des tâches aux différents rôles
- Un système de contrôle d'accès contrôlant l'instanciation des rôles par des sujets
- Ce dernier doit faire le pont entre des rôles organisationnels (réceptionniste), et des rôles au sein du BP

Contrôle d'accès dans les processus métiers (1/2)

Principe

- Chaque BP doit définir un certain nombre de rôles
- Les tâches sont réparties entre ces rôles
- Il faut vérifier que les personnes qui exécutent les tâches peuvent effectivement être actives dans les rôles qui sont définis

Workflow et contrôle d'accès

- un diagramme d'exécution, avec une distribution des tâches aux différents rôles
- Un système de contrôle d'accès contrôlant l'instanciation des rôles par des sujets
- Ce dernier doit faire le pont entre des rôles organisationnels (réceptionniste), et des rôles au sein du BP

RÉCURSIVITÉ

QU'EST-CE QU'UNE TÂCHE ?

exemple : accueil d'un patient

- Il s'agit d'un élément d'un BP (accueillir un patient est une tâche dans le BP définissant la réaction en cas d'accident)
- Mais c'est aussi souvent un BP. Pour accueillir un patient :
  - il faut réserver un lit
  - récupérer (si c'est possible), son identité, puis, s'il existe, son dossier médical
  - l'envoyer en salle d'examen
  - ...
- Si les tâches sont aussi des BP, elles sont aussi protégées par un système de contrôle d'accès

QU'EST-CE QU'UNE TÂCHE ?

exemple : accueil d'un patient

- Il s'agit d'un élément d'un BP (accueillir un patient est une tâche dans le BP définissant la réaction en cas d'accident)
- Mais c'est aussi souvent un BP. Pour accueillir un patient :
  - il faut réserver un lit
  - récupérer (si c'est possible), son identité, puis, s'il existe, son dossier médical
  - l'envoyer en salle d'examen
  - ...
- Si les tâches sont aussi des BP, elles sont aussi protégées par un système de contrôle d'accès

RÉCURSIVITÉ

QU'EST-CE QU'UNE TÂCHE ?

exemple : accueil d'un patient

- Il s'agit d'un élément d'un BP (accueillir un patient est une tâche dans le BP définissant la réaction en cas d'accident)
- Mais c'est aussi souvent un BP. Pour accueillir un patient :
  - il faut réserver un lit
  - récupérer (si c'est possible), son identité, puis, s'il existe, son dossier médical
  - l'envoyer en salle d'examen
  - ...
- Si les tâches sont aussi des BP, elles sont aussi protégées par un système de contrôle d'accès

RÉCURSIVITÉ

QU'EST-CE QU'UNE TÂCHE ?

exemple : accueil d'un patient

- Il s'agit d'un élément d'un BP (accueillir un patient est une tâche dans le BP définissant la réaction en cas d'accident)
- Mais c'est aussi souvent un BP. Pour accueillir un patient :
  - il faut réserver un lit
  - récupérer (si c'est possible), son identité, puis, s'il existe, son dossier médical
  - l'envoyer en salle d'examen
  - ...
- Si les tâches sont aussi des BP, elles sont aussi protégées par un système de contrôle d'accès

QU'EST-CE QU'UNE TÂCHE ?

exemple : accueil d'un patient

- Il s'agit d'un élément d'un BP (accueillir un patient est une tâche dans le BP définissant la réaction en cas d'accident)
- Mais c'est aussi souvent un BP. Pour accueillir un patient :
  - il faut réserver un lit
  - récupérer (si c'est possible), son identité, puis, s'il existe, son dossier médical
  - l'envoyer en salle d'examen
  - ...
- Si les tâches sont aussi des BP, elles sont aussi protégées par un système de contrôle d'accès

CONTRÔLE D'ACCÈS DANS LES PROCESSUS MÉTIERS (2/2)

Morale :

- Les tâches au sein d'un BP ne peuvent être effectuées par un sujet que si certaines conditions sont remplies
- Pouvoir jouer un rôle ne suffit pas nécessairement
- Il faut aussi pouvoir définir les permissions, c'est-à-dire les conditions sous-lesquelles une tâche (une action) peut être effectuée

$\Rightarrow$ mécanisme de contrôle intégré à l'application ou pare-feu

EXEMPLE : LOP

Loan Origination Process

- Processus métier du domaine bancaire
- Rôles impliqués : pre-clerk, post-clerk, manager, client, credit rating bureau
- Contraintes de contrôle d'accès :
  - Pour les trois premiers rôles, avoir un poste spécifique (rôle organisationnel) au sein de la banque
  - Le CRB peut être joué par un organisme agréé
  - Contraintes métiers :
    - C'est seulement un manager qui peut s'occuper des gros prêts
    - lorsque des clerks sont impliqués, les pre- et post-clerks doivent être joués par deux employés différents
    - Les données de crédit du client ne doivent pas être fournies à la banque

Différents niveaux d'application de la politique

Exemple d'implémentation

- Vérification des rôles "bancaires" : pare-feu applicatif intégré au serveur
- Vérification des contraintes métier : intégrées à l'application

Autre possibilité :

- Données de l'application stockées dans une base externe
- Cette base est consultée par un pare-feu extérieur à l'application

OUTLINE

APPLICATIONS WEB

Authentification, Attribut, Autorisation (1/2)

Authentication, Attribute, Authorization

Modèle en couches

- Couche Authentification : Un principal se voit attribuer une identité par une autorité appelée *serveur d'identité*
- Couche Attribut : (plus ou moins optionnelle) à partir de l'identité fournie par la couche d'authentification, des autorités décident des attributs que possède le principal ayant l'identité
- Couche Autorisation : En fonction des attributs qui sont attachés à l'identité d'un client, délivre les tokens permettant l'accès à un ou plusieurs service

Authentification, Attribut, Autorisation (2/2)

Exemple

- Authentification : login nom/mot de passe
- Attribut : le principal obtient un token disant qu'il est actif dans un rôle *r*
- Autorisation : le principal obtient des tokens représentant des permissions en fonction du rôle dans lequel il est actif

Les couches servent de guide pour structurer le contrôle d'accès.

COUCHE D'AUTHENTIFICATION

notion centrale : le serveur d'identité

- Toute entité (*principal*) est connectée à un serveur d'identité
- Ce serveur authentifie les principaux suivant des méthodes déclarées :
  - PKI/Cryptographie
  - Couple username/password
  - Mécanismes physiques
- Il fournit au principal un ou plusieurs certificats décrivant :
  - Son identité dans le système
  - La méthode d'authentification choisie
- Anonymat : cette identité peut être à long ou à court terme (nonce)

Exemple de serveur d'identité : Shibboleth

COUCHE D'ATTRIBUTS

notion centrale : négociation de la confiance

- Un client initie une négociation en commençant par négocier son identité
- Lorsque l'identité est établie par un serveur d'identité, une séquence d'échange d'assertions commence
- Lorsque le client a obtenu suffisament d'assertions, il arrête la négociation (nonce)

Exemple de serveur d'identité : Shibboleth

COUCHE D'ATTRIBUTS

notion centrale : négociation de la confiance

- Un client initie une négociation en commençant par négocier son identité
- Lorsque l'identité est établie par un serveur d'identité, une séquence d'échange d'assertions commence
- Lorsque le client a obtenu suffisament d'assertions, il arrête la négociation

INTUITION

- L'échange de certificats correspond à un calcul de plus petit point fixe
- C'est la partie "clause" des politiques de contrôle d'accès
- Pour les transactions courtes, il suffit en général d'avoir le username pour autoriser l'accès, donc cette étape est sautée

COUCHE D'ATTRIBUTS

notion centrale : négociation de la confiance

- Un client initie une négociation en commençant par négocier son identité
- Lorsque l'identité est établie par un serveur d'identité, une séquence d'échange d'assertions commence
- Lorsque le client a obtenu suffisament d'assertions, il arrête la négociation

INTUITION

- L'échange de certificats correspond à un calcul de plus petit point fixe
- C'est la partie "clause" des politiques de contrôle d'accès
- Pour les transactions courtes, il suffit en général d'avoir le username pour autoriser l'accès, donc cette étape est sautée

Couche d'autorisation

notions principales : PDP/PEP

Point de Décision de la Politique

- Autorité qui décide des accès à un service en fonction :
  - des attributs du requêrant
  - de l'état de l'application
- Si l'accès est autorisé, un token d'autorisation est envoyé au principal, qui peut l'utiliser auprès du service

En pratique :

- Le PDP peut être intégré à l'application ou à un pare-feu indépendant
- Sauf dans les cas complexes, on passe directement des attributs aux autorisations
- Cas complexes : agilité/souplesse de la politique du service avec des niveaux d'indirection dans l'expression du CA

PLAN

Outline

Contrôle d'accès pour les services Web

SAML

Requêtes XACML

PRÉSENTATION GÉNÉRALE

SAML

- Langage permettant d'exprimer des assertions
- Ces assertions peuvent exprimer :
  - un attribut d'un sujet (ex : le rôle dans lequel il est actif)
  - une autorisation (permission de faire une action sur un objet)
- Initialement développé pour le *Single Sign On* (identité fédérée par SUN
- Maintenant adopté par tout le monde WS

Présentation générale

SAML

- Langage permettant d'exprimer des assertions
- Ces assertions peuvent exprimer :
  - un attribut d'un sujet (ex : le rôle dans lequel il est actif)
  - une autorisation (permission de faire une action sur un objet)
- Initialement développé pour le *Single Sign On* (identité fédérée par SUN
- Maintenant adopté par tout le monde WS

Présentation générale

SAML

- Langage permettant d'exprimer des assertions
- Ces assertions peuvent exprimer :
  - un attribut d'un sujet (ex : le rôle dans lequel il est actif)
  - une autorisation (permission de faire une action sur un objet)
- Initialement développé pour le *Single Sign On* (identité fédérée par SUN
- Maintenant adopté par tout le monde WS

Présentation générale

SAML

- Langage permettant d'exprimer des assertions
- Ces assertions peuvent exprimer :
  - un attribut d'un sujet (ex : le rôle dans lequel il est actif)
  - une autorisation (permission de faire une action sur un objet)
- Initialement développé pour le *Single Sign On* (identité fédérée par SUN
- Maintenant adopté par tout le monde WS

PRÉSENTATION GÉNÉRALE

SAML

- Langage permettant d'exprimer des assertions
- Ces assertions peuvent exprimer :
  - un attribut d'un sujet (ex : le rôle dans lequel il est actif)
  - une autorisation (permission de faire une action sur un objet)
- Initialement développé pour le *Single Sign On* (identité fédérée par SUN
- Maintenant adopté par tout le monde WS

PRÉSENTATION GÉNÉRALE

SAML

- Langage permettant d'exprimer des assertions
- Ces assertions peuvent exprimer :
  - un attribut d'un sujet (ex : le rôle dans lequel il est actif)
  - une autorisation (permission de faire une action sur un objet)
- Initialement développé pour le *Single Sign On* (identité fédérée par SUN
- Maintenant adopté par tout le monde WS

“Single Sign-on”

- on veut éviter les identifications à répétitions des utilisateurs du Web
- problème encore plus crucial dans le cas de services Web (communications intensives)
- Single Sign-on : le premier site ayant authentifié un agent certifie son identité auprès d’autres sites pendant une certaine durée
- on (IBM,Microsoft) parle aussi de problème d’identité fédérée

“Single Sign-on”

- on veut éviter les identifications à répétitions des utilisateurs du Web
- problème encore plus crucial dans le cas de services Web (communications intensives)
- Single Sign-on : le premier site ayant authentifié un agent certifie son identité auprès d’autres sites pendant une certaine durée
- on (IBM,Microsoft) parle aussi de problème d’identité fédérée

“Single Sign-on”

- on veut éviter les identifications à répétitions des utilisateurs du Web
- problème encore plus crucial dans le cas de services Web (communications intensives)
- Single Sign-on : le premier site ayant authentifié un agent certifie son identité auprès d’autres sites pendant une certaine durée
- on (IBM,Microsoft) parle aussi de problème d’identité fédérée

“Single Sign-on”

- on veut éviter les identifications à répétitions des utilisateurs du Web
- problème encore plus crucial dans le cas de services Web (communications intensives)
- Single Sign-on : le premier site ayant authentifié un agent certifie son identité auprès d’autres sites pendant une certaine durée
- on (IBM,Microsoft) parle aussi de problème d’identité fédérée

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

GÉNÉRALITÉS

- But : gérer les identités d'agents
- principe de base : une autorité émet une affirmation (*statement*) sur un sujet à une date donnée
- Ce quadruplet est en général signé, et certaines parties peuvent être chiffrées
- Il peut aussi y être associé des conditions (*e.g.* date limite de validité) en dehors desquelles l'affirmation n'est plus assurée
- L'affirmation peut porter sur une identification, une autorisation ou un attribut du sujet
- note : on verra plus loin que c'est ce qui est demandé par XACML !
- Le chiffrement et la signature sont définis par XML-ENC et XML-SIG
- on ne rentre pas dans les détails. . .

Interprétation logique de SAML

SAML	$\leftrightarrow$Logique
certificat	$\leftrightarrow$entrée dans une BD
ensemble des certificats	$\leftrightarrow$BD distribuée
PDP	$\leftrightarrow$ensemble de règles

En plus, il est nécessaire de garder la trace de l'émetteur (*issuer*) d'un certificat pour savoir s'il peut être pris en compte (= si le PDP l'ajoute dans sa base de faits) lorsqu'un utilisateur demande un accès

Important

SAML permet de réaliser une base de donnée distribuée pour le contrôle d'accès

OUTLINE

CONTRÔLE D'ACCÈS POUR LES SERVICES WEB

SAML

Requêtes XACML

Requêtes et réponses

- le PEP reçoit une demande d'accès à une ressource.
- cette demande peut contenir des jetons SAML, des références LDAP, etc.
- le PEP formule, d'après cette demande, une requête auprès du PDP
- cette requête n'est pas forcément une requête XACML, mais le PDP doit être capable dans tous les cas de la transformer en requête XACML
- après consultation des autorités compétentes, le PDP envoie une réponse XACML au PEP
- le PEP interprète cette décision, et peut soit la notifier tout de suite à l'émetteur de la requête, soit consulter d'autres PDP en cas d'échec

Requêtes et réponses

- le PEP reçoit une demande d'accès à une ressource.
- cette demande peut contenir des jetons SAML, des références LDAP, etc.
- le PEP formule, d'après cette demande, une requête auprès du PDP
- cette requête n'est pas forcément une requête XACML, mais le PDP doit être capable dans tous les cas de la transformer en requête XACML
- après consultation des autorités compétentes, le PDP envoie une réponse XACML au PEP
- le PEP interprète cette décision, et peut soit la notifier tout de suite à l'émetteur de la requête, soit consulter d'autres PDP en cas d'échec

Requêtes et réponses

- le PEP reçoit une demande d'accès à une ressource.
- cette demande peut contenir des jetons SAML, des références LDAP, etc.
- le PEP formule, d'après cette demande, une requête auprès du PDP
- cette requête n'est pas forcément une requête XACML, mais le PDP doit être capable dans tous les cas de la transformer en requête XACML
- après consultation des autorités compétentes, le PDP envoie une réponse XACML au PEP
- le PEP interprète cette décision, et peut soit la notifier tout de suite à l'émetteur de la requête, soit consulter d'autres PDP en cas d'échec

Requêtes et réponses

- le PEP reçoit une demande d'accès à une ressource.
- cette demande peut contenir des jetons SAML, des références LDAP, etc.
- le PEP formule, d'après cette demande, une requête auprès du PDP
- cette requête n'est pas forcément une requête XACML, mais le PDP doit être capable dans tous les cas de la transformer en requête XACML
- après consultation des autorités compétentes, le PDP envoie une réponse XACML au PEP
- le PEP interprète cette décision, et peut soit la notifier tout de suite à l'émetteur de la requête, soit consulter d'autres PDP en cas d'échec

Requêtes et réponses

- le PEP reçoit une demande d'accès à une ressource.
- cette demande peut contenir des jetons SAML, des références LDAP, etc.
- le PEP formule, d'après cette demande, une requête auprès du PDP
- cette requête n'est pas forcément une requête XACML, mais le PDP doit être capable dans tous les cas de la transformer en requête XACML
- après consultation des autorités compétentes, le PDP envoie une réponse XACML au PEP
- le PEP interprète cette décision, et peut soit la notifier tout de suite à l'émetteur de la requête, soit consulter d'autres PDP en cas d'échec

Requêtes et réponses

- le PEP reçoit une demande d'accès à une ressource.
- cette demande peut contenir des jetons SAML, des références LDAP, etc.
- le PEP formule, d'après cette demande, une requête auprès du PDP
- cette requête n'est pas forcément une requête XACML, mais le PDP doit être capable dans tous les cas de la transformer en requête XACML
- après consultation des autorités compétentes, le PDP envoie une réponse XACML au PEP
- le PEP interprète cette décision, et peut soit la notifier tout de suite à l'émetteur de la requête, soit consulter d'autres PDP en cas d'échec

Requêtes

- identification du sujet
- action
- objet de l'action
- contexte de la requête

Requêtes

- identification du sujet
- action
- objet de l'action
- contexte de la requête

Requêtes

- identification du sujet
- action
- objet de l'action
- contexte de la requête

Requêtes

- identification du sujet
- action
- objet de l'action
- contexte de la requête

DÉFINITION D'ENTITÉS

- Permet d'alléger les notations
- Raccourcis remplacés avant le traitement en tant que document XML
- Avec la déclaration :

```
<!ENTITY xml
"http://www.w3.org/2001/XMLSchema#">
```

- on peut noter :

```
&xml;string
```

au lieu de :

```
http://www.w3.org/2001/XMLSchema#string
```

DÉFINITION D'ENTITÉS

- Permet d'alléger les notations
- Raccourcis remplacés avant le traitement en tant que document XML
- Avec la déclaration :

```
<!ENTITY xml
"http://www.w3.org/2001/XMLSchema#">
```

- on peut noter :

```
&xml;string
```

au lieu de :

```
http://www.w3.org/2001/XMLSchema#string
```

Définition d'entités

- Permet d'alléger les notations
- Raccourcis remplacés avant le traitement en tant que document XML
- Avec la déclaration :

```
<!ENTITY xml
"http://www.w3.org/2001/XMLSchema#">
```

- on peut noter :

```
&xml;string
```

  au lieu de :

```
http://www.w3.org/2001/XMLSchema#string
```

Définition d'entités

- Permet d'alléger les notations
- Raccourcis remplacés avant le traitement en tant que document XML
- Avec la déclaration :

```
<!ENTITY xml
"http://www.w3.org/2001/XMLSchema#">
```

- on peut noter :

```
&xml;string
```

au lieu de :

```
http://www.w3.org/2001/XMLSchema#string
```

ENTITÉS DANS LA SUITE DU COURS

```
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY xacml "urn:oasis:names:tc:xacml:">
<!ENTITY rcombine "&xacml;1.0:rule-combining-algorithm:">
<!ENTITY pcombine "&xacml;1.0:policy-combining-algorithm:"
<!ENTITY function "&xacml;1.0:function:">
<!ENTITY subject-category "&xacml;1.0:subject-category:">
<!ENTITY subject "&xacml;1.0:subject:">
<!ENTITY context "&xacml;1.0:context">
<!ENTITY role "&xacml;2.0:subject:role">
<!ENTITY roles "urn:example:role-values:">
<!ENTITY resource "&xacml;1.0:resource:">
<!ENTITY action "&xacml;1.0:action:">
<!ENTITY actions "&xacml;2.0:actions:">
<!ENTITY environment "&xacml;1.0:environment:">
<!ENTITY datatype "&xacml;1.0:data-type:">
<!ENTITY status "&xacml;1.0:status">
```

ATTRIBUTS

- exemple

```
<Attribute
  AttributeId="&subject;subject-id"
  DataType="&datatype;rfc822Name">
  <AttributeValue>
    boss@example.org
  </AttributeValue>
</Attribute>
```

- tag : Attribute
- attributs :
  - AttributeID : le type de l'attribut dans la politique de contrôle d'accès
  - DataType : le type `xml:Schema` de l'attribut
- Un ou plusieurs fils AttributeValue

ATTRIBUTS

- exemple

```
<Attribute
  AttributeId="&subject;subject-id"
  DataType="&datatype;rfc822Name">
  <AttributeValue>
    boss@example.org
  </AttributeValue>
</Attribute>
```

- tag : Attribute
- attributs :
  - AttributeID : le type de l'attribut dans la politique de contrôle d'accès
  - DataType : le type `xml:Schema` de l'attribut
- Un ou plusieurs fils AttributeValue

ATTRIBUTS

- exemple

```
<Attribute
  AttributeId="&subject;subject-id"
  DataType="&datatype;rfc822Name">
  <AttributeValue>
    boss@example.org
  </AttributeValue>
</Attribute>
```

- tag : Attribute
- attributs :
  - AttributeID : le type de l'attribut dans la politique de contrôle d'accès
  - DataType : le type `xml:Schema` de l'attribut
- un ou plusieurs fils AttributeValue

ATTRIBUTS

- exemple

```
<Attribute
  AttributeId="&subject;subject-id"
  DataType="&datatype;rfc822Name">
  <AttributeValue>
    boss@example.org
  </AttributeValue>
</Attribute>
```

- tag : Attribute
- attributs :
  - AttributeID : le type de l'attribut dans la politique de contrôle d'accès
  - DataType : le type `xml:Schema` de l'attribut
- Un ou plusieurs fils AttributeValue

ATTRIBUTS

- exemple

```
<Attribute
  AttributeId="&subject;subject-id"
  DataType="&datatype;rfc822Name">
  <AttributeValue>
    boss@example.org
  </AttributeValue>
</Attribute>
```

- tag : Attribute
- attributs :
  - AttributeID : le type de l'attribut dans la politique de contrôle d'accès
  - DataType : le type `xml:Schema` de l'attribut
- Un ou plusieurs fils AttributeValue

ATTRIBUTS

- exemple

```
<Attribute
  AttributeId="&subject;subject-id"
  DataType="&datatype;rfc822Name">
  <AttributeValue>
    boss@example.org
  </AttributeValue>
</Attribute>
```

- tag : Attribute
- attributs :
  - AttributeID : le type de l'attribut dans la politique de contrôle d'accès
  - DataType : le type `xml:Schema` de l'attribut
- Un ou plusieurs fils AttributeValue

ATTRIBUTS (SCHÉMA)

```
<xs:element name="Attribute"
  type="xacml-context:AttributeType"/>
<xs:complexType name="AttributeType">
  <xs:sequence>
    <xs:element
          ref="xacml-context:AttributeValue"
          maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="AttributeId"
       type="xs:anyURI" use="required"/>
  <xs:attribute name="DataType"
       type="xs:anyURI" use="required"/>
  <xs:attribute name="Issuer"
       type="xs:string" use="optional"/>
</xs:complexType>
```

AttributeId sert à identifier l'attribut auquel on va donner une valeur. Un attribut peut avoir plusieurs valeurs possibles (ex. groups dans LDAP)

VALEURS D'ATTRIBUTS (SCHÉMA)

```
<xs:element name="AttributeValue"
      type="xacml-context:AttributeValueType"/>
<xs:complexType name="AttributeValueType" mixed="true
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any"
      processContents="lax"/>
</xs:complexType>
```

Les valeurs possibles des attributs dépendent de l'élément (sujet, resource,etc.) qu'ils qualifient. Ce ne sont pas des attributs au sens XML du terme.

EXEMPLE – REQUÊTE

```
<Request xmlns="&context;"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instanc
  <Subject>
    ...
  </Subject>
  <Resource>
    ...
  </Resource>
  <Action>
    ...
  </Action>
</Request>
```

Requête (Schéma)

```
<xs:element name="Request"
     type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:Subject" maxOccurs="unbounded
    <xs:element
      ref="xacml-context:Resource" maxOccurs="unbounde
    <xs:element
      ref="xacml-context:Action"/>
    <xs:element
      ref="xacml-context:Environment"/>
  </xs:sequence>
</xs:complexType>
```

SUJET (EXEMPLE)

```
<Subject>
 <Attribute AttributeId="&xacml;2.0:subject-category"
    DataType="&xml;anyURI"
    Issuer="admin@example.org">
      <AttributeValue>
        &xacml;1.0:subject-category:access-subject
      </AttributeValue>
 </Attribute>
</Subject>
```

SUJET (SCHÉMA)

```
<xs:element name="Subject"
      type="xacml-context:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:Attribute"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute
    name="SubjectCategory" type="xs:anyURI"
    default="&xacml;1.0:subject-category:access-subje
</xs:complexType>
```

SUJETS (COMMENTAIRES)

Les sujets peuvent être de différentes catégories suivant le statut du sujet dans le cadre de la requête :

- access-subject : sujet qui a initié la requête
- recipient-subject : sujet qui recevra le résultat de la requête
- intermediary-subject : sujet qui sert d'intermédiaire dans la requête (plusieurs possibles)
- codebase : sujet lorsqu'il s'agit d'une requête demandée par un système d'information
- requesting-machine : identité la machine d'où est issue la requête

Sujets (commentaires)

Les sujets peuvent être de différentes catégories suivant le statut du sujet dans le cadre de la requête :

- access-subject : sujet qui a initié la requête
- recipient-subject : sujet qui recevra le résultat de la requête
- intermediary-subject : sujet qui sert d'intermédiaire dans la requête (plusieurs possibles)
- codebase : sujet lorsqu'il s'agit d'une requête demandée par un système d'information
- requesting-machine : identité la machine d'où est issue la requête

Sujets (commentaires)

Les sujets peuvent être de différentes catégories suivant le statut du sujet dans le cadre de la requête :

- access-subject : sujet qui a initié la requête
- recipient-subject : sujet qui recevra le résultat de la requête
- intermediary-subject : sujet qui sert d'intermédiaire dans la requête (plusieurs possibles)
- codebase : sujet lorsqu'il s'agit d'une requête demandée par un système d'information
- requesting-machine : identité la machine d'où est issue la requête

Sujets (commentaires)

Les sujets peuvent être de différentes catégories suivant le statut du sujet dans le cadre de la requête :

- access-subject : sujet qui a initié la requête
- recipient-subject : sujet qui recevra le résultat de la requête
- intermediary-subject : sujet qui sert d'intermédiaire dans la requête (plusieurs possibles)
- codebase : sujet lorsqu'il s'agit d'une requête demandée par un système d'information
- requesting-machine : identité la machine d'où est issue la requête

Sujets (commentaires)

Les sujets peuvent être de différentes catégories suivant le statut du sujet dans le cadre de la requête :

- access-subject : sujet qui a initié la requête
- recipient-subject : sujet qui recevra le résultat de la requête
- intermediary-subject : sujet qui sert d'intermédiaire dans la requête (plusieurs possibles)
- codebase : sujet lorsqu'il s'agit d'une requête demandée par un système d'information
- requesting-machine : identité la machine d'où est issue la requête

Sujets (commentaires)

Il est possible d'utiliser les types de données simples définis dans XMLSchema ou des types définis dans XACML :

- x500name : structure contenant des données relatives à un utilisateur (email, pays,etc.)
- rfc822name : adresse email
- ipAdresse : adresse IP
- dnsName : nom Web (www.google.fr)

Sujets (commentaires)

Il est possible d'utiliser les types de données simples définis dans XMLSchema ou des types définis dans XACML :

- x500name : structure contenant des données relatives à un utilisateur (email, pays,etc.)
- rfc822name : adresse email
- ipAdresse : adresse IP
- dnsName : nom Web (www.google.fr)

Sujets (commentaires)

Il est possible d'utiliser les types de données simples définis dans XMLSchema ou des types définis dans XACML :

- x500name : structure contenant des données relatives à un utilisateur (email, pays,etc.)
- rfc822name : adresse email
- ipAdresse : adresse IP
- dnsName : nom Web (www.google.fr)

Sujets (commentaires)

Il est possible d'utiliser les types de données simples définis dans XMLSchema ou des types définis dans XACML :

- x500name : structure contenant des données relatives à un utilisateur (email, pays,etc.)
- rfc822name : adresse email
- ipAdresse : adresse IP
- dnsName : nom Web (www.google.fr)

Sujets (commentaires)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

SUJETS (COMMENTAIRES)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

SUJETS (COMMENTAIRES)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

Sujets (commentaires)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

Sujets (commentaires)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

SUJETS (COMMENTAIRES)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

Sujets (commentaires)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

Sujets (commentaires)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

Sujets (commentaires)

Les attributs sont importants, car le contrôle d'accès se fonde sur ces valeurs pour former une décision

- subject-category : type du sujet (voir 2 planches plus haut)
- subject-id : nom (type string) du sujet. Un autre format peut être précisé avec un type de la planche précédente
- subject-id-qualifier : cet attribut indique le serveur qui administre l'identité de sujet de la requête
- key-info : indique la clef publique utilisée pour vérifier l'identité du sujet
- authentication-time et authn-locality:authentication-method : date et méthode utilisée pour l'authentification
- request-time : indique l'heure à laquelle la requête a été formulée
- start-session-time : heure à laquelle la session a débuté

Il est aussi possible d'utiliser les attributs LDAP ou SAML

RESSOURCES (EXEMPLE)

```
<Resource>
  <Attribute AttributeId="&resource;resource-id"
    DataType="&xml;anyURI">
      <AttributeValue>
        http://www.example.org/res.html
      </AttributeValue>
  </Attribute>
</Resource>
```

Ressources (schéma)

```
<xs:element name="Resource"
     type="xacml-context:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:ResourceContent" minOccurs="0
    <xs:element
      ref="xacml-context:Attribute"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

`ResourceContent` est un nœud pouvant contenir n'importe quoi

RESSOURCES (COMMENTAIRES)

- resource-id : identifie la resource
- target-namespace : ns de l'élément fils de ResourceContent

Ressources (commentaires)

- resource-id : identifie la resource
- target-namespace : ns de l'élément fils de ResourceContent

ACTION (SCHÉMA)

```
<xs:element name="Action"
     type="xacml-context:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:Attribute"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Actions (commentaires)

Plusieurs attributs possibles

- action-id : identificateur de l'action. Cet identificateur doit avoir un sens dans le ns action-namespace (qui doit être défini)
- implied-action : attribut à positionner lorsque l'action peut être déduite du contexte d'exécution

Actions (commentaires)

Plusieurs attributs possibles

- action-id : identificateur de l'action. Cet identificateur doit avoir un sens dans le ns action-namespace (qui doit être défini)
- implied-action : attribut à positionner lorsque l'action peut être déduite du contexte d'exécution

Environnement (schéma)

```
<xs:element name="Environment"
     type="xacml-context:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:Attribute"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

EXEMPLE – RÉPONSE

```
<Response>
  <Result ResourceID=
    "http://www.example.org/private/privatepage.html"
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value=
    "&status;ok"/>
    </Status>
    <Obligations>
      ...
    </Obligations>
  </Result>
</Response>
```

Réponse (schéma)

```
<xs:element name="Result"
     type="xacml-context:ResultType"/>
<xs:complexType name="ResultType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:Decision"/>
    <xs:element
      ref="xacml-context:Status" minOccurs="0"/>
    <xs:element
      ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute
     name="ResourceId" type="xs:string" use="optiona
</xs:complexType>
```

`ResourceId` permet d'identifier la ressource sur laquelle s'applique la décision

Réponse (commentaires)

- Une réponse encapsule le résultat du PDP envoyé au PEP
- Elle peut contenir plusieurs résultats en cas de requête multiple
- Les requêtes multiples sont optionnelles
- Le résultat s'applique à la ressource donnée dans l'attribut optionnel de Result si elle est précisé

Réponse (commentaires)

- Une réponse encapsule le résultat du PDP envoyé au PEP
- Elle peut contenir plusieurs résultats en cas de requête multiple
- Les requêtes multiples sont optionnelles
- Le résultat s'applique à la ressource donnée dans l'attribut optionnel de Result si elle est précisé

Réponse (commentaires)

- Une réponse encapsule le résultat du PDP envoyé au PEP
- Elle peut contenir plusieurs résultats en cas de requête multiple
- Les requêtes multiples sont optionnelles
- Le résultat s'applique à la ressource donnée dans l'attribut optionnel de Result si elle est précisé

Réponse (commentaires)

- Une réponse encapsule le résultat du PDP envoyé au PEP
- Elle peut contenir plusieurs résultats en cas de requête multiple
- Les requêtes multiples sont optionnelles
- Le résultat s'applique à la ressource donnée dans l'attribut optionnel de Result si elle est précisé

DÉCISION (SCHÉMA)

```
<xs:element name="Decision"
     type="xacml-context:DecisionType"/>
<xs:simpleType name="DecisionType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Permit"/>
    <xs:enumeration value="Deny"/>
    <xs:enumeration value="Indeterminate"/>
    <xs:enumeration value="NotApplicable"/>
  </xs:restriction>
</xs:simpleType>
```

Décision

- Permit : la requête est accepté sous réserve que les obligations soient remplies
- Deny : la politique refuse l'accès à la ressource
- Indeterminate : il n'est pas possible d'appliquer la politique à la requête, parce qu'un élément (sujet,etc.) est inconnu ou parce que la construction de la politique ne permet pas d'aboutir à une décision (erreur)
- NotApplicable : il n'est pas possible d'appliquer la politique à la requête parce que le service n'est pas compétent pour répondre

Décision

- Permit : la requête est accepté sous réserve que les obligations soient remplies
- Deny : la politique refuse l'accès à la ressource
- Indeterminate : il n'est pas possible d'appliquer la politique à la requête, parce qu'un élément (sujet,etc.) est inconnu ou parce que la construction de la politique ne permet pas d'aboutir à une décision (erreur)
- NotApplicable : il n'est pas possible d'appliquer la politique à la requête parce que le service n'est pas compétent pour répondre

Décision

- Permit : la requête est accepté sous réserve que les obligations soient remplies
- Deny : la politique refuse l'accès à la ressource
- Indeterminate : il n'est pas possible d'appliquer la politique à la requête, parce qu'un élément (sujet,etc.) est inconnu ou parce que la construction de la politique ne permet pas d'aboutir à une décision (erreur)
- NotApplicable : il n'est pas possible d'appliquer la politique à la requête parce que le service n'est pas compétent pour répondre

Décision

- Permit : la requête est accepté sous réserve que les obligations soient remplies
- Deny : la politique refuse l'accès à la ressource
- Indeterminate : il n'est pas possible d'appliquer la politique à la requête, parce qu'un élément (sujet,etc.) est inconnu ou parce que la construction de la politique ne permet pas d'aboutir à une décision (erreur)
- NotApplicable : il n'est pas possible d'appliquer la politique à la requête parce que le service n'est pas compétent pour répondre

STATUS (SCHÉMA)

```
<xs:element name="Status"
     type="xacml-context:StatusType"/>
<xs:complexType name="StatusType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:StatusCode"/>
    <xs:element
      ref="xacml-context:StatusMessage" minOccurs="0",
    <xs:element
      ref="xacml-context:StatusDetail" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

STATUS (SUITE DU SCHÉMA)

```
<xs:element name="StatusCode"
     type="xacml-context:StatusCodeType"/>
<xs:complexType name="StatusCodeType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:StatusCode" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute
     name="Value" type="xs:anyURI" use="required"/>
</xs:complexType>
```

STATUS (SUITE DU SCHÉMA)

```
<xs:element name="StatusMessage"
     type="xs:string"/>
<xs:element name="StatusDetail"
     type="xacml-context:StatusDetailType"/>
<xs:complexType name="StatusDetailType">
  <xs:sequence>
    <xs:any namespace="##any"
      processContents="lax"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Erreur

```
<xs:element name="MissingAttributeDetail"
      type="xacml-context:MissingAttributeDetailType"/>
<xs:complexType name="MissingAttributeDetailType">
  <xs:sequence>
    <xs:element
      ref="xacml-context:AttributeValue"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute
      name="AttributeId" type="xs:anyURI"
      use="required"/>
  <xs:attribute
      name="DataType" type="xs:anyURI"
      use="required"/>
  <xs:attribute
      name="Issuer" type="xs:string"
      use="optional"/>
</xs:complexType>
```

Élément optionel en général, et ne doit pas être inclus dans une requête dont le statut est ok

OBLIGATIONS

- Les obligations sont à remplir par le PEP
- exemple : tracer la requête dans un log
- Si le PEP ne comprend pas ou ne peut pas accomplir l'obligation, la requête est considérée comme refusée (deny)
- Cette partie est définie selon l'implémentation

OBLIGATIONS

- Les obligations sont à remplir par le PEP
- exemple : tracer la requête dans un log
- Si le PEP ne comprend pas ou ne peut pas accomplir l'obligation, la requête est considérée comme refusée (deny)
- Cette partie est définie selon l'implémentation

OBLIGATIONS

- Les obligations sont à remplir par le PEP
- exemple : tracer la requête dans un log
- Si le PEP ne comprend pas ou ne peut pas accomplir l'obligation, la requête est considérée comme refusée (deny)
- Cette partie est définie selon l'implémentation

OBLIGATIONS

- Les obligations sont à remplir par le PEP
- exemple : tracer la requête dans un log
- Si le PEP ne comprend pas ou ne peut pas accomplir l'obligation, la requête est considérée comme refusée (deny)
- Cette partie est définie selon l'implémentation