

# Module RSX112 - Sécurité des Réseaux

Stéphane LARCHER



# 1. Introduction - Fondements de la Sécurité

1

## La Triade CIA Revisitée

Dans le contexte du contrôle d'accès, la triade CIA prend une dimension opérationnelle :

- Confidentialité : Seules les entités autorisées accèdent aux informations
- Intégrité : Protection contre les modifications non autorisées
- Disponibilité : Accès garanti aux ressources pour les utilisateurs légitimes

2

## Principes Fondamentaux

Principe de moindre privilège : Un utilisateur ne doit avoir que les droits minimaux nécessaires à l'accomplissement de ses tâches.

Séparation des tâches : Les opérations critiques sont divisées entre plusieurs personnes pour éviter les fraudes.

Defense in depth : Plusieurs couches de sécurité se complètent.

3

## Terminologie Essentielle

- Sujet : Entité active (utilisateur, processus) qui demande l'accès
- Objet : Ressource passive (fichier, base de données) à protéger
- Référence Monitor : Composant qui contrôle tous les accès
- TCB (Trusted Computing Base) : Ensemble des composants critiques pour la sécurité

## 2. Authentification

### 2.1 Définition et Enjeux

L'authentification est le processus permettant de vérifier l'identité revendiquée d'un sujet. Elle constitue la première ligne de défense dans tout système de contrôle d'accès.

### 2.2 Facteurs d'Authentification

Quelque chose que vous connaissez (Something you know)

Mots de passe : La méthode la plus répandue mais problématique.

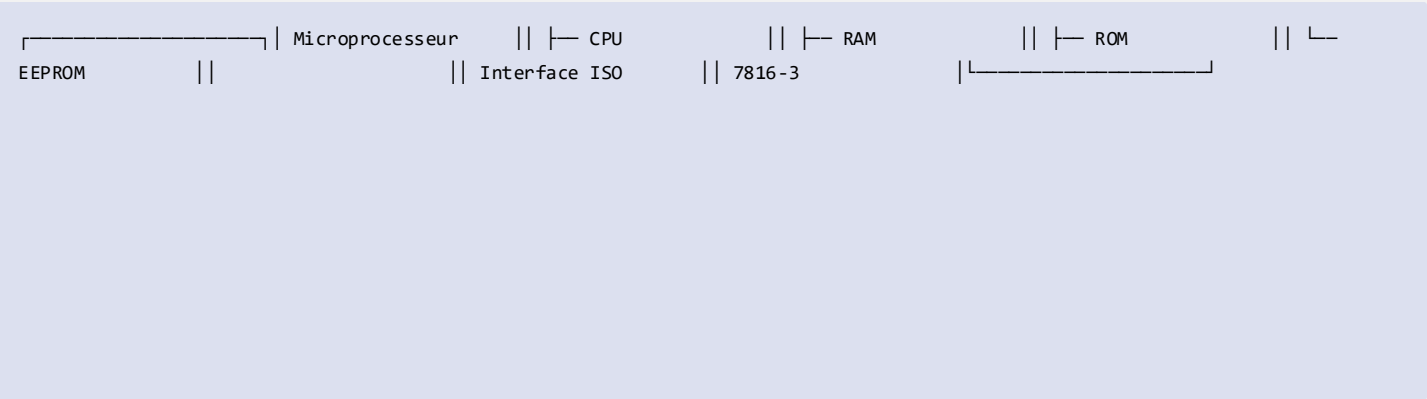
Techniques de stockage sécurisé :

# Techniques de stockage sécurisé des mots de passe

```
# Stockage naïf (à ne JAMAIS faire)password_plaintext = "motdepasse123"# Hachage simple (insuffisant)import hashlib
password_hash = hashlib.sha256("motdepasse123".encode()).hexdigest()# Hachage avec sel (recommandé)import bcrypt
password = "motdepasse123"
salt = bcrypt.gensalt(rounds=12)
hashed = bcrypt.hashpw(password.encode('utf-8'), salt)# Vérificationbcrypt.checkpw(password.encode('utf-8'), hashed)
```

Cartes à puce :

Structure carte à puce :



Protocole d'authentification carte :

1. Insertion : Détection physique et electrical reset
2. ATR (Answer To Reset) : Carte envoie ses paramètres
3. SELECT : Sélection de l'application
4. VERIFY : Présentation du PIN
5. Challenge-Response : Authentification cryptographique

Fonctions de dérivation de clés :

- PBKDF2 : Standard PKCS#5, itératif avec sel
- bcrypt : Basé sur Blowfish, résistant aux attaques par GPU
- scrypt : Résistant aux attaques parallèles (memory-hard)
- Argon2 : Gagnant du concours Password Hashing Competition

Exemple de configuration bcrypt :

```
# Configuration recommandée
2024rounds = 12 # ~250ms sur CPU moderne
salt = bcrypt.gensalt(rounds=rounds)
```

Quelque chose que vous possédez (Something you have)

Jetons (Tokens) :

- Hardware tokens : RSA SecurID, YubiKey
- Software tokens : Applications TOTP (Google Authenticator, Authy)
- SMS/Email OTP : Moins sûr (SIM swapping, interception)

Quelque chose que vous êtes (Something you are)

Biométrie physiologique :

- Empreintes digitales : Minuties (bifurcations, terminaisons)
- Reconnaissance de l'iris : Texture complexe, 240+ points distinctifs
- Reconnaissance faciale : Géométrie faciale, algorithmes deep learning
- Géométrie de la main : Longueur des doigts, largeur

Biométrie comportementale :

- Frappe au clavier : Rythme, pression, temps entre touches
- Dynamique de signature : Pression, vitesse, accélération
- Démarche : Pattern de marche unique

Métriques biométriques :

- FAR (False Accept Rate) : Taux de fausses acceptations
- FRR (False Reject Rate) : Taux de faux rejets
- EER (Equal Error Rate) : Point d'équilibre FAR = FRR

Qualité biométrique :

EER < 1% : Excellente (iris)

EER 1-5% : Bonne (empreinte)



# Génération et vérification TOTP

```
# Génération secret partagésecret = pyotp.random_base32()# TOTP (30s window)totp = pyotp.TOTP(secret)current_otp = totp.now()
```

# Vérification TOTP et WebAuthn

```
# Vérification avec windowis_valid = totp.verify(user_otp, valid_window=1)
```

WebAuthn/FIDO2 : Standard moderne sans mot de passe

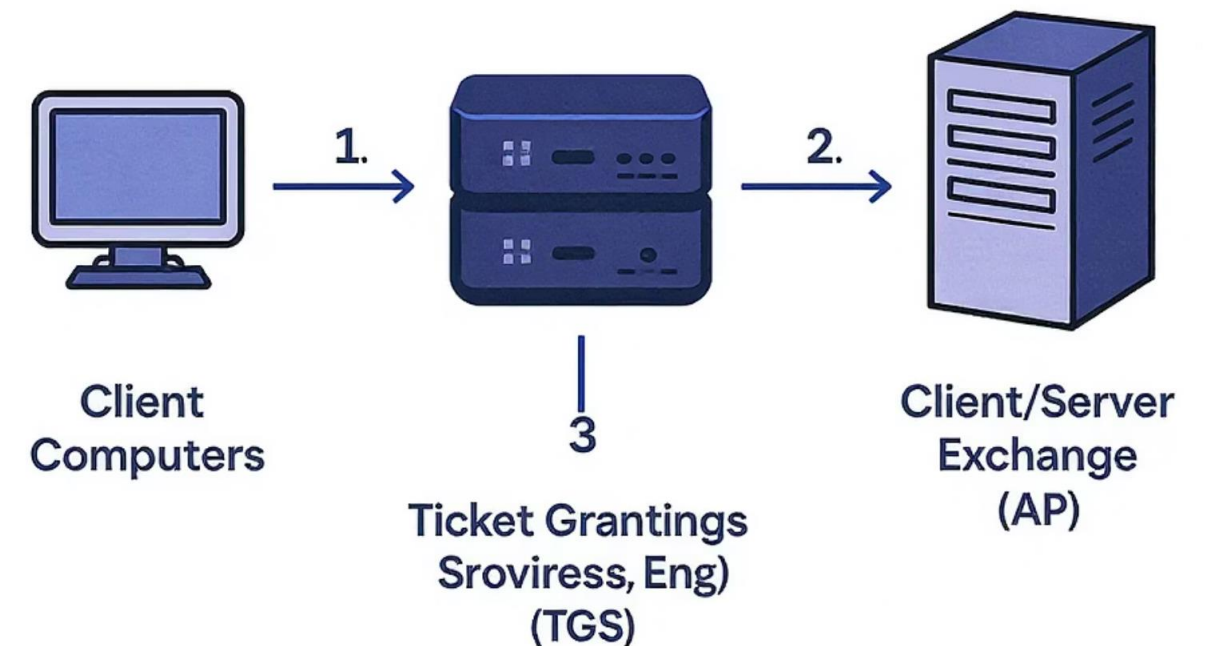
```
// Enregistrementnavigator.credentials.create({ publicKey: { challenge:  
challenge, rp: {name: "Example Corp"}, user: {id: userId, name:  
"alice@example.com"}, pubKeyCredParams: [{alg: -7, type: "public-key"}],  
authenticatorSelection: { userVerification: "required" } }));
```

## 2.4 Protocoles d'Authentification Distribués

### Kerberos

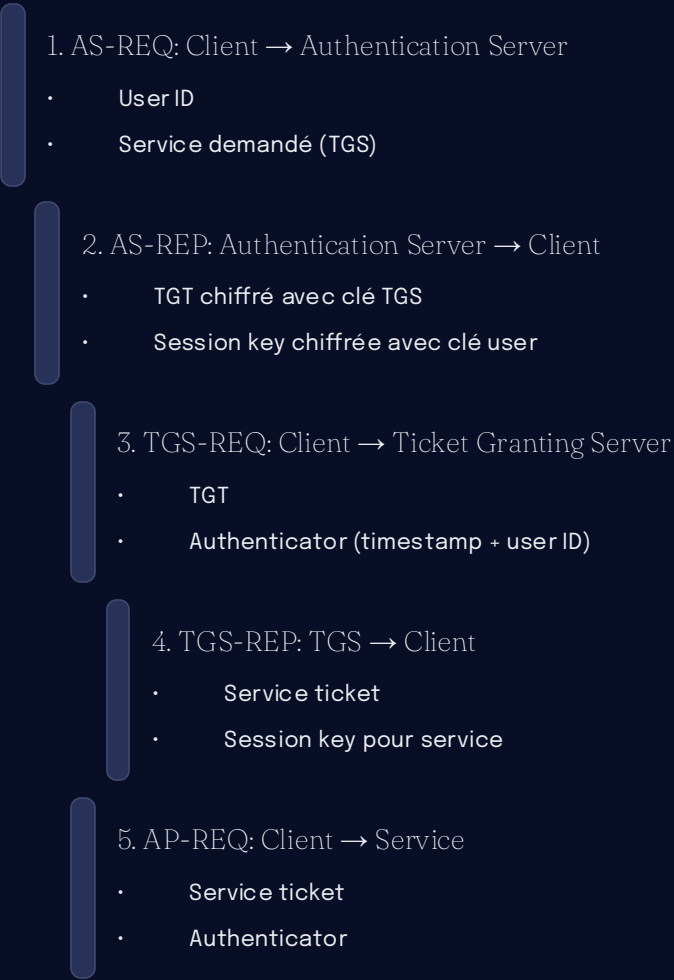
Architecture basée sur la cryptographie symétrique :

## KERBEROS AUTHENTICATION PROTOCOL





# Protocoles d'Authentification Distribués



SAML 2.0

Protocole de fédération d'identités :

```
saml:Response saml:Assertion saml:Subject saml:NameID alice@corp.com saml:AttributeStatement  saml:AttributeValue Engineering
```

# 6. Autorisation et Modèles de Sécurité

## 3.1 Mécanismes d'Autorisation

### Access Control Lists (ACL)

Structure classique associant sujets et permissions à chaque objet :

```
Fichier: /home/alice/document.txtACL:|-- alice: read, write, execute|-- bob:
read|-- group:staff: read|-- other: none
```

## ACCESS CONTROL LIST FILE PERMISSIONS

✓ Contetuation

Escrparry

File Permissions

● AT





# Représentation numérique et ACL étendues

```
# Représentation numérique Unixrwxr--r-- = 744 (octal)
```

ACL étendues (POSIX.1e):

```
# Ajout d'une entrée ACLsetfacl -m u:bob:rw file.txt# ACL par défaut pour répertoiresetfacl -d -m g:developers:rwx /project# Visualisationgetfacl file.txt
```

## Capabilities

Approche alternative : chaque sujet possède une liste de capacités

```
Processus PID 1234:Capabilities:|—  
CAP_READ_FILE:/etc/passwd|— CAP_WRITE_FILE:/tmp/*|—  
CAP_NET_BIND_SERVICE:port<1024|— CAP_SYS_TIME (réglage horloge)
```

Linux Capabilities :

# Linux Capabilities

```
# Affichage des capabilities d'un processus
cat /proc/1234/status | grep Cap# Attribution de capability spécifiques
setcap cap_net_bind_service=+ep /usr/bin/nginx# Exécution avec capabilities limitées
capsh --drop=cap_sys_admin --chroot=/jail -- -c "command"
```

## 3.2 Modèles de Sécurité Hiérarchiques

### Modèle Bell-LaPadula (Confidentialité)

Propriétés fondamentales :

- Simple Security (ss-property) : "No read up"
  - Un sujet de niveau L ne peut lire un objet de niveau > L
- Star Property (\*-property) : "No write down"
  - Un sujet de niveau L ne peut écrire dans un objet de niveau < L

Niveaux de classification :

TOP SECRET (4) ↑

SECRET (3) ↑

CONFIDENTIEL (2) ↑

PUBLIC (1)

Exemple d'application :

Alice (SECRET) peut :

✓ Lire: PUBLIC, CONFIDENTIEL, SECRET

X Lire: TOP SECRET

✓ Écrire: SECRET, TOP SECRET

X Écrire: PUBLIC, CONFIDENTIEL

### Modèle Biba (Intégrité)

Dual de Bell-LaPadula pour l'intégrité :

Propriétés :

- Simple Integrity : "No read down"
- Star Integrity : "No write up"

Niveaux d'intégrité :

CRUCIAL (4) - Données critiques système

HIGH (3) - Données vérifiées

MEDIUM (2) - Données standard

LOW (1) - Données non vérifiées

## Modèles à Compartiments

Extension avec catégories orthogonales aux niveaux :

Classification complète : (Niveau, {Compartiments})

# Modèles à Compartiments - Exemples

Exemples de classifications

- (SECRET, {NATO, CRYPTO})
- (CONFIDENTIEL, {NUCLEAR})
- (TOP SECRET, {INTEL, WEAPONS, CRYPTO})

Relation de dominance

$$(L1, C1) \text{ domine } (L2, C2) \Leftrightarrow L1 \geq L2 \text{ et } C2 \subseteq C1$$

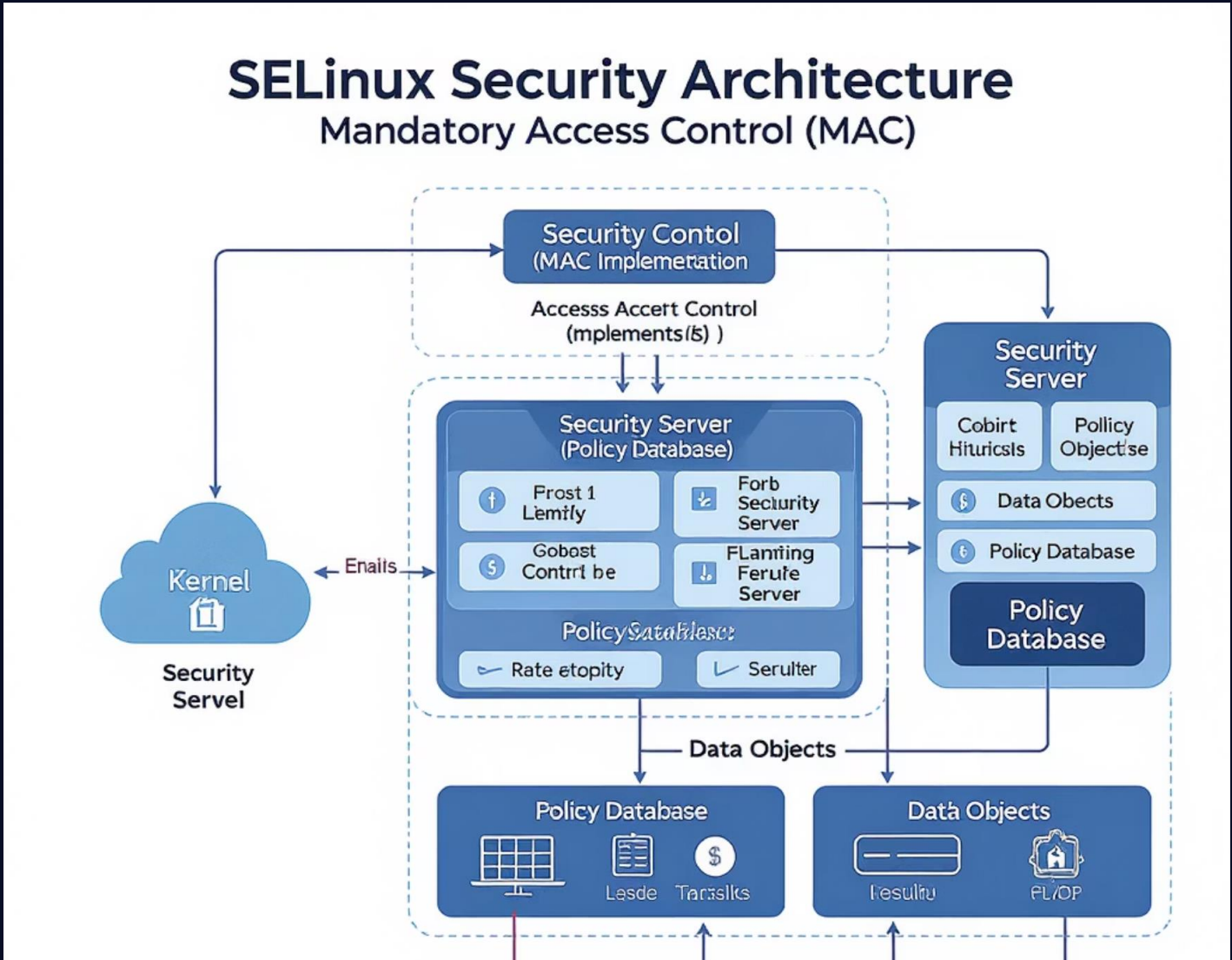
## 3.3 Implémentations Modernes

SELinux (Security-Enhanced Linux)

Architecture :

Application↓Kernel syscall↓DAC check (traditional Unix)↓MAC check (SELinux) ← Policy Engine↓Access granted/denied

Contextes de sécurité :



# SELinux - Contextes et Politiques

```
# Format: user:role:type:level -Z /home/alice/-rw-r--r--. alice unconfined_u:object_r:user_home_t:s0 file.txt#  
Processusps -eZ | grep httpdunconfined_u:system_r:httpd_t:s0 1234 ? httpd
```

Types de politiques :

- Targeted : Protection ciblée (services réseau)
- Strict : Tout est contrôlé par MAC
- MLS : Multi-Level Security (classifications)

Règles de politique :

# SELinux - Règles de politique

```
# Allow httpd to read web contentallow httpd_t httpd_exec_t:file { read execute };# Transition de domainetype_transition init_t  
httpd_exec_t:process httpd_t;# Contrainte MLSmlsconstrain file write (l1 eq l2);
```

Windows Mandatory Access Control

Niveaux d'intégrité Windows :

- System (0x3000) - Système
- High (0x2000) - Administrateur
- Medium (0x1000) - Utilisateur standard
- Low (0x0000) - Processus restreint

Exemple d'application :

# Windows MAC - Exemple d'application

```
# Création d'un processus en intégrité faible$startInfo = New-Object  
System.Diagnostics.ProcessStartInfo$startInfo.UseShellExecute =  
$false$startInfo.FileName = "notepad.exe"$startInfo.Arguments = "/low"
```

## 3.4 Politiques Discretionnaires vs Obligatoires

### DAC (Discretionary Access Control)

- Principe : Le propriétaire contrôle les accès
- Flexibilité : Haute, delegation possible
- Exemples : Unix permissions, Windows ACL
- Faiblesse : Programmes Trojan, escalade de privilèges

### MAC (Mandatory Access Control)

- Principe : Politique centralisée, non modifiable
- Contrôle : Strict, appliqué par le système
- Exemples : SELinux, Trusted Solaris
- Avantage : Protection contre code malveillant





# 1. Classification et Gestion des Accès

## 4.1 Classification CIA selon FIPS 199

Catégories d'Impact	Moderate Impact	High Impact
<b>Low Impact : Perte limitée, effet mineur</b> <ul style="list-style-type: none"><li>Dégradation des opérations</li><li>Dommages financiers mineurs</li><li>Préjudice personnel limité</li></ul>	<b>Perte significative</b> <ul style="list-style-type: none"><li>Dégradation majeure des opérations</li><li>Dommages financiers importants</li><li>Préjudice personnel sérieux</li></ul>	<b>Perte grave ou catastrophique</b> <ul style="list-style-type: none"><li>Incapacité à remplir les missions</li><li>Dommages financiers majeurs</li><li>Préjudice grave à la sécurité</li></ul>

### Matrice de Classification

Système d'information : CRM Enterprise

	C	I	A
Données clients	M	H	L
Config système	L	H	M
Logs d'audit	M	H	L

Classification système : H (plus haut niveau)

## 4.2 Standards ISO 27000

### ISO 27001 - Exigences

Contrôles d'accès (Annexe A.9) :

- A.9.1 Exigences métier pour le contrôle d'accès
- A.9.2 Gestion des accès utilisateur
- A.9.3 Responsabilités utilisateur
- A.9.4 Contrôle d'accès aux systèmes et applications

### ISO 27002 - Bonnes Pratiques

Politique de contrôle d'accès :

# Politique de contrôle d'accès

1 Identification des exigences métier  
Déterminer les besoins d'accès en fonction des processus métier et des responsabilités des utilisateurs.

2 Classification de l'information  
Catégoriser les données selon leur sensibilité et leur importance pour l'organisation.

3 Définition des profils d'accès  
Créer des modèles d'accès standardisés basés sur les rôles et les responsabilités.

4 Procédures de gestion des droits  
Établir des processus formels pour l'attribution, la modification et la révocation des droits d'accès.

5 Révision périodique des accès  
Mettre en place un calendrier de vérification régulière des droits attribués pour s'assurer qu'ils restent appropriés.

## 4.3 Contrôle d'Accès Basé sur les Rôles (RBAC)

### Modèle RBAC

Utilisateurs ↔ Rôles ↔ Permissions ↔ Objets

# Modèle RBAC - Exemples

Exemple :

Alice ← Comptable →  
{Lecture\_Factures,  
Écriture\_Comptes}

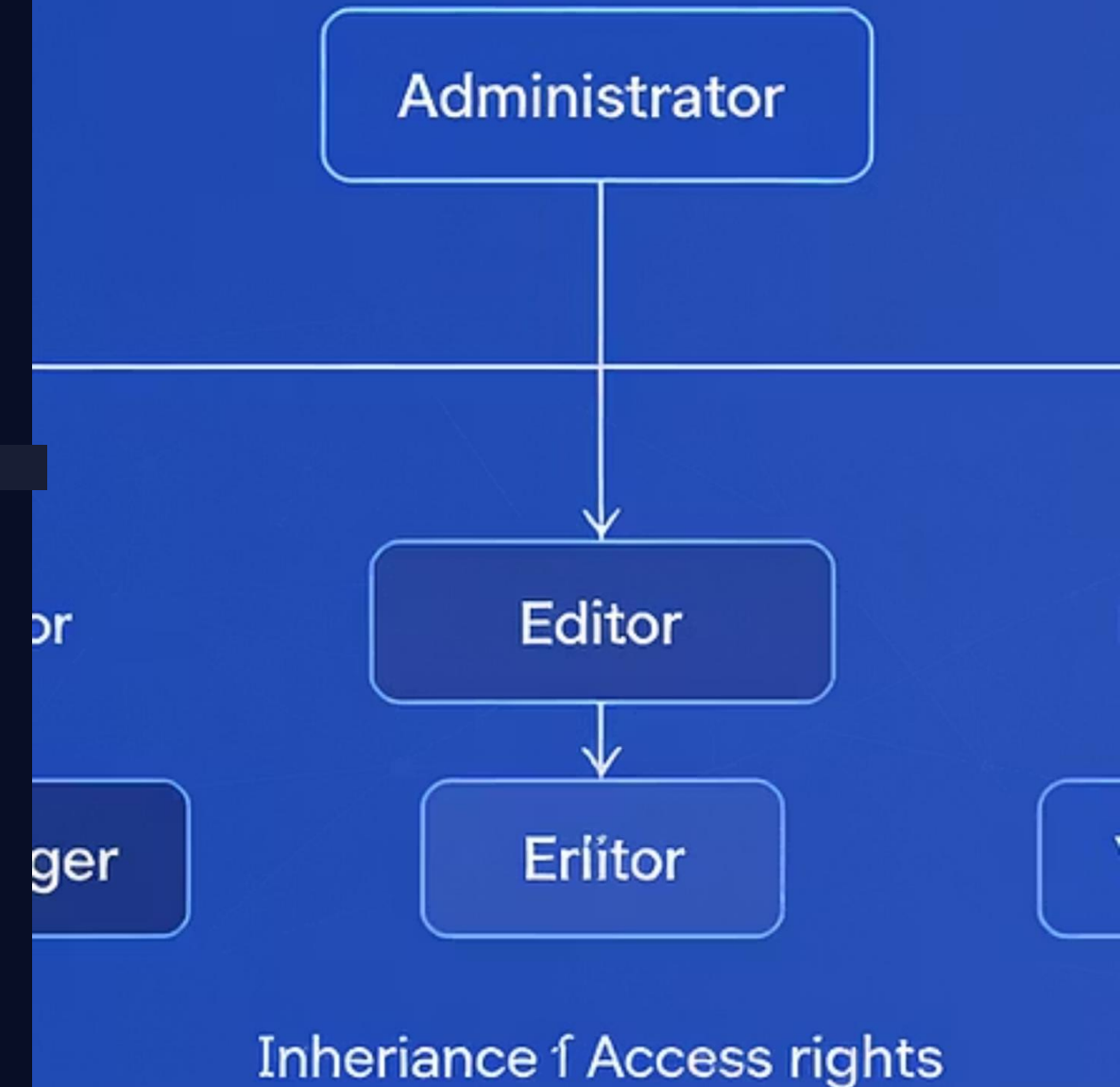
Bob ← Manager →  
{Lecture\_Rapports,  
Validation\_Budgets}

Hiérarchie de rôles :

```
Directeur_Général    ↓  
Directeur_IT        ↓  
↓Administrateur_Systeme  
Chef_Projet         ↓  
↓   Technicien  
Développeur
```

RBAC dans Active Directory

## e-Based Access Control



# RBAC dans Active Directory

```
# Création d'un groupe de sécuritéNew-ADGroup -Name "Finance_ReadOnly" -GroupScope Global# Attribution de permissions sur GPOSet-GPPermissions -Name "Finance_Policy" -TargetName "Finance_ReadOnly" -PermissionLevel GpoRead# Ajout utilisateur au rôle
```

# RBAC dans Active Directory (suite)

```
Add-ADGroupMember -Identity "Finance_ReadOnly" -Members "alice"
```

## 4.4 Gestion des Identités Privilégiées



### Comptes à Privilèges

#### Types de comptes :

- Comptes de service : Applications, démons
- Comptes d'administration : root, Administrator
- Comptes d'urgence : Break-glass accounts
- Comptes partagés : À éviter absolument



### PAM (Privileged Access Management)

#### Fonctionnalités essentielles :

# PAM - Fonctionnalités essentielles

## 1. Coffre-fort des mots de passe

- Rotation automatique
- Checkout/Checkin
- Audit complet

## 2. Proxy de connexion

- Session recording
- Commandes interdites
- Contrôle temps réel



# PAM - Fonctionnalités essentielles (suite)

## 3. Élévation de privilèges

- Just-in-time access
- Workflow d'approbation
- Durée limitée

Exemple de politique PAM :

```
policy: name: "Linux_Root_Access" target: "production_servers" conditions:
```

# GED ACCESS MANA



Evaluest is  
Exaluated



Granted v  
Denied

## Politique PAM - Conditions

```
- business_hours: true - approval_required: true - max_duration:  
"4h" actions:
```

# Politique PAM - Actions

```
- record_session: true - notify_admin: true
```

# Politique PAM - Actions (suite)

```
- forbidden_commands: ["rm -rf", "dd", "format"]
```

## 4.5 Principe de Séparation des Tâches

### Contrôles Compensatoires

Séparation statique : Rôles mutuellement exclusifs

Règle :  $\forall u \in \text{Utilisateurs}, \neg(\text{Comptable}(u) \wedge \text{Auditeur}(u))$



## Séparation des Tâches - Implémentation

```
Implémentation :if user.hasRole("Comptable"):  
user.denyRole("Auditeur")
```

Séparation dynamique : Contraintes d'exécution

Politique : Une transaction > 10k€ nécessite 2 approbations

# Séparation dynamique - Workflow

Demandeur  $\neq$  Approbateur1

La personne qui initie la transaction ne peut pas être la première à l'approuver.

Approbateur1  $\neq$  Approbateur2

Les deux approbateurs doivent être des personnes différentes pour garantir un contrôle dual.

Niveaux hiérarchiques appropriés

Tous les intervenants doivent avoir le niveau hiérarchique requis pour leur rôle dans le processus.

## Exemples Sectoriels

Banque (SOX Compliance)

- Séparation initiation/validation des virements
- Contrôle développement/production
- Audit indépendant des accès

Pharmaceutique (FDA 21 CFR Part 11)

- Signature électronique à deux personnes
- Séparation recherche/qualité
- Traçabilité complète des modifications



# 4. Canaux Cachés et Contrôle d'Inférence

## 5.1 Canaux Cachés (Covert Channels)

### Définition et Classification

Canal caché : Mécanisme de communication non prévu par la politique de sécurité, permettant à deux processus de communiquer de manière dissimulée.

Types de canaux :

- Stockage : Information encodée dans l'état du système
- Timing : Information encodée dans la temporisation

### Covert\_TCP - Exemple Pratique

Principe : Utilisation des champs TCP pour encoder des données

```
// Covert_TCP - Canal caché via IP IDstruct iphdr
*ip_header = (struct iphdr *)packet;
```

# Covert\_TCP - Exemple Pratique

```
// Émission : encoder data dans IP IDip_header->id = htons(secret_byte);  
// Réception : extraire data depuis IP  
IDsecret_byte = ntohs(ip_header->id);
```

Canaux via TCP :

1. IP ID field : 16 bits par paquet
2. TCP sequence : 32 bits, mais détectable
3. TCP window : 16 bits, variations naturelles
4. TCP options : Champs timestamp, padding
5. TCP flags : Combinaisons non standard

Autres Techniques de Canaux Cachés

Canal système de fichiers :

# Canal caché via système de fichiers

```
# Émission : créer/supprimer fichiers selon pattern  
echo "1" > /tmp/.bit1  
# bit 1  
rm /tmp/.bit0          # bit 0  
# Réception : tester existence  
if [ -f /tmp/.bit1 ]; then    bit=1; else    bit=0; fi
```

Canal temporel :

```
import time
```

# Canal caché temporel

```
# Émissiondef send_bit(bit):    if bit == 1:        time.sleep(0.1)    # Long délai = 1    else:        time.sleep(0.05)    # Court délai = 0# Réceptiondef receive_bit():    start = time.time()    # Observer temps de réponse du système    return 1    if response_time > 0.075 else 0
```

Détection et Contre-mesures

Analyse statistique :



# Détection de canaux cachés - Analyse statistique

```
# Détection d'anomalies dans les timingsimport numpy as npfrom scipy import stats
```

# Détection de canaux cachés - Analyse statistique (suite)

```
def detect_timing_channel(response_times): # Test de normalité stat, p_value = stats.shapiro(response_times) #  
Coefficient de variation anormal cv = np.std(response_times) / np.mean(response_times) return p_value < 0.05 or cv >   
threshold
```



# Contre-mesures pour canaux cachés

## Mitigation

- Normalisation des délais : Padding temporel
- Bruit artificiel : Injection d'aléa
- Limitation de bande passante : Rate limiting
- Monitoring : Détection d'anomalies

## 5.2 Contrôle d'Inférence dans les Bases de Données

### Problématique

Les bases de données statistiques permettent des requêtes agrégées tout en préservant la confidentialité individuelle. Cependant, des attaques par inférence peuvent révéler des informations privées.

### Types d'Attaques par Inférence

Attaque par différence :

```
-- Révéler le salaire de Alice  
SELECT AVG(salary) FROM  
employees WHERE department='IT';  
-- Résultat : 55,000€
```

# Attaque par différence (suite)

```
SELECT AVG(salary) FROM employees WHERE department='IT' AND name != 'Alice';-- Résultat : 52,000€
```

# Attaque par différence (résultat)

```
-- Inférence : Salaire Alice  $\approx 3 \times 55,000 - 2 \times 52,000 = 61,000\text{€}$ 
```

Attaque par recoupement :

```
-- Requête 1SELECT COUNT(*) FROM patients WHERE age BETWEEN 40 AND 50 AND disease='diabetes';-- Résultat : 23--  
Requête 2SELECT COUNT(*) FROM patients WHERE age BETWEEN 40 AND 50 AND disease='diabetes' AND city='Paris';--  
Résultat : 1
```

# Attaque par recoupement (résultat)

```
-- Inférence : Un seul diabétique 40-50 ans à Paris
```

Techniques de Protection

Seuils de restriction :

```
MIN_GROUP_SIZE = 5
```



# Seuils de restriction - Implémentation

```
def execute_query(sql_query): result_count = count_affected_records(sql_query) if result_count < MIN_GROUP_SIZE:  
    return "Query denied - insufficient group size" return execute(sql_query)
```

# Bruit différentiel (Differential Privacy)

```
import numpy as np
def add_laplace_noise(true_value, sensitivity, epsilon): """ epsilon : budget de confidentialité
    (plus petit = plus privé) sensitivity : changement max causé par un individu """
    scale = sensitivity / epsilon
    noise = np.random.laplace(0, scale)
    return true_value + noise
```

# Bruit différentiel - Exemple

```
# Exempletrue_average = 55000 # Moyenne réelle des salairesnoisy_average = add_laplace_noise(true_average, 10000, 0.1)
```

Généralisation des données :

Données originales :

Age: 34, Salaire: 45,000€, Code postal: 75001





# Généralisation des données

Données originales :

Age: 34, Salaire: 45,000€, Code postal: 75001

Données généralisées :

Age: [30-40], Salaire: [40k-50k], Zone: Paris Centre

k-anonymat :

Principe : Chaque individu est indistinguishable de  $k-1$  autres



# k-anonymat - Table originale

Table originale :Age Genre Code Postal Maladie23 M 75001 Diabetes24 M 75001 Diabetes25 F 75002 HypertensionTable 2-anonyme :





# k-anonymat - Table 2-anonyme

Age Genre Zone Maladie20-30 \* 75001-2 Diabetes20-30 \* 75001-2 Diabetes20-30 \* 75001-2 Hypertension





# 1. Synthèse et Perspectives

## 6.1 Points Clés à Retenir





### Authentification Moderne

-  MFA obligatoire pour les accès privilégiés
-  Migration vers WebAuthn/FIDO2
-  Biométrie comme facteur complémentaire, pas unique
-  Élimination progressive des mots de passe

### Autorisation Granulaire

-  RBAC avec hiérarchies et contraintes temporelles
-  Principe de moindre privilège strictement appliqué
-  Séparation des tâches automatisée
-  Zero Trust Architecture en émergence

### Modèles de Sécurité

-  MAC obligatoire pour les systèmes critiques
-  Classification automatisée par IA
-  Étiquetage de sécurité bout-en-bout
-  Adaptation aux environnements cloud

## 6.2 Tendances Émergentes

### Zero Trust Security Model

Principe : "Never trust, always verify"

Verification,  
Access,



# Zero Trust Security Model - Architecture

Architecture :

Device	Policy	Resource	Verification
Enforcement	Access	Engine	Point
Control			

Adaptive Access Control

# Contrôle d'accès adaptatif

# Contrôle d'accès adaptatif basé sur le risque

# Calcul du score de risque

```
def calculate_risk_score(user, resource, context): risk = 0 # Facteurs utilisateur risk += user.recent_failed_logins * 10
risk += user.privilege_level * 5 # Facteurs contextuels if context.location != user.usual_location: risk += 20 if
context.time not in user.usual_hours: risk += 15 if context.device != user.registered_devices: risk += 25 # Facteurs
ressource risk += resource.classification_level * 10 return min(risk, 100)
```

# Décision d'accès adaptative

```
def adaptive_access_decision(risk_score):    if risk_score < 30:        return "ALLOW"    elif risk_score < 60:  
return "ALLOW_WITH_MFA"    elif risk_score < 80:        return "ALLOW_WITH_APPROVAL"    else:        return "DENY"
```

Privacy-Preserving Analytics

Homomorphic Encryption pour les calculs sur données chiffrées :



# Calcul sur données chiffrées

```
# Pseudocode - Calcul sur données chiffrées
```



# Homomorphic Encryption - Exemple

```
encrypted_salaries = [encrypt(salary) for salary in salaries]
encrypted_sum = homomorphic_add(encrypted_salaries)
encrypted_average = homomorphic_divide(encrypted_sum, len(encrypted_salaries))
average = decrypt(encrypted_average) # Seul le résultat est déchiffré
```

## 6.3 Défis Futurs

### Quantique et Post-Quantique

- Menace : Algorithmes de Shor/Grover sur crypto classique
- Réponse : Migration vers crypto post-quantique
- Timeline : 2025-2030 pour migration complète

### IA et Machine Learning

- Opportunités : Détection d'anomalies, classification automatique
- Risques : Attaques adversariales, biais algorithmiques
- Besoin : Frameworks de gouvernance IA

### Conformité Réglementaire

- RGPD : Privacy by design
- NIS2 : Sécurité des infrastructures critiques
- AI Act : Gouvernance de l'IA
- Cyber Resilience Act : Sécurité des produits connectés

## 6.4 Recommandations Pratiques

1. Audit régulier : Révision trimestrielle des droits d'accès
2. Formation utilisateurs : Sensibilisation aux risques
3. Monitoring continu : SIEM avec corrélation d'événements
4. Test d'intrusion : Validation des contrôles
5. Plan de continuité : Procédures d'urgence documentées

# Questions de révision

- 1 Expliquez la différence entre authentification et autorisation
- 2 Pourquoi le modèle Bell-LaPadula interdit-il l'écriture vers le bas ?
- 3 Comment détecter un canal caché temporel ?
- 4 Quels sont les avantages et inconvénients du k-anonymat ?
- 5 Comment implémenter la séparation des tâches dans un système RBAC ?

Module RSX112 - Contrôle d'Accès et Sécurité de l'Information

CNAM - 2024-2025