

# Intégration d'une politique de flot de contrôle dans un automate de sécurité

Guillaume Bouffard, Mathieu Lassale, Sergio Ona Domene, **Hanan Tadmori** et Jean-Louis Lanet

Smart Secure Devices (SSD) Team, Université de Limoges

[hanan.tadmori@xlim.fr](mailto:hanan.tadmori@xlim.fr)

**SARSSI- 2013**  
**16-18 Septembre 2013**

# Plan de la présentation

**Contexte général**

**Problèmes de sécurité**

**État de l'art**

**Approche adoptée**

- **Implémentation**
- **Métriques**

**Conclusion**

# Carte à puce

Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

- Microprocesseur
- Mémoires ROM, EEPROM et RAM
- Industries des télécommunications, bancaire, audiovisuelle, secteur du gouvernement
- Java Card **ORACLE®**



# Sources des problèmes de sécurité

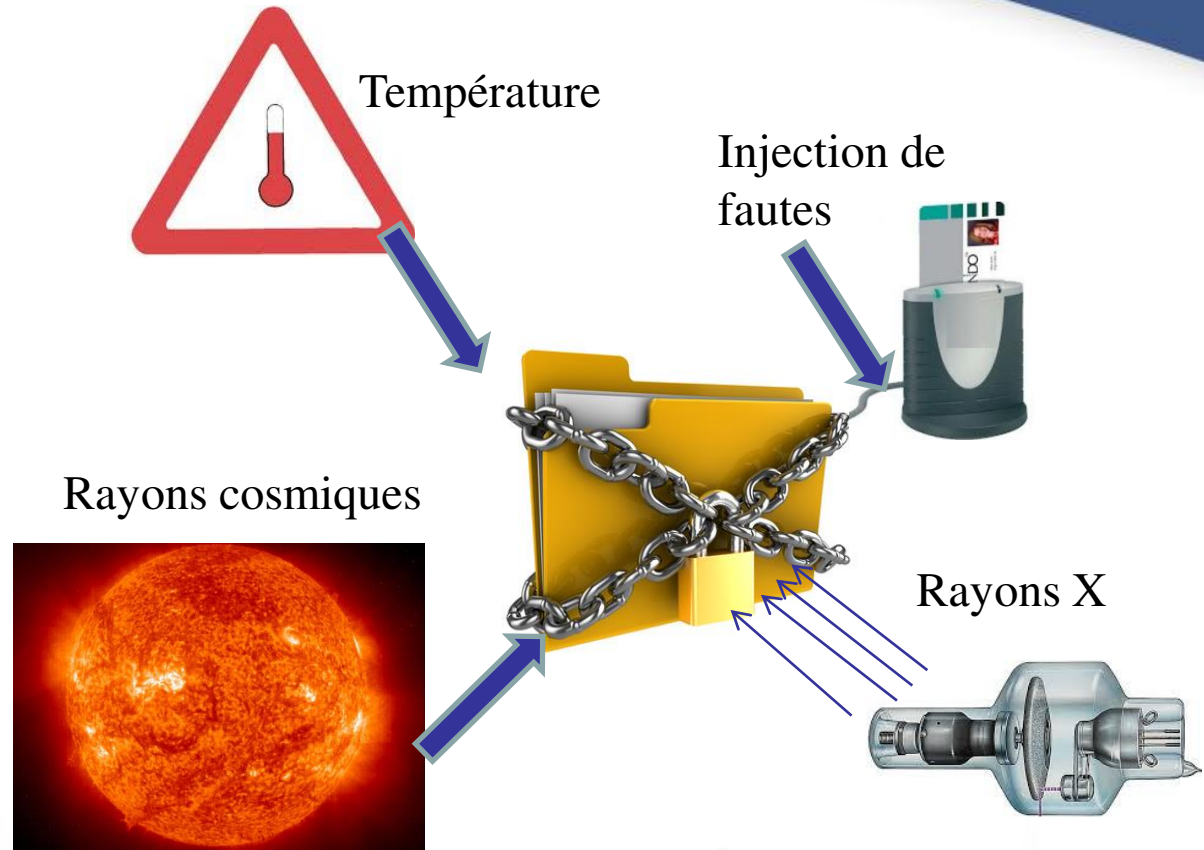
Contexte général

**Problèmes de sécurité**

État de l'art

Approche adoptée

Conclusion



Intégration d'une politique de flot de contrôle dans un automate de sécurité

# Attaques par injection de fautes

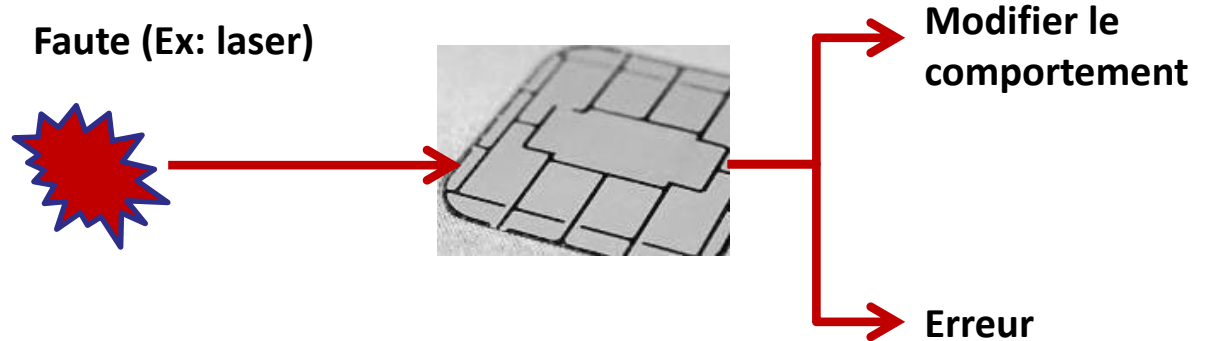
Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion



- Perturber le microprocesseur, les mémoires, les bus ...
- Comportement permanent /transitoire

# Attaques par injection de fautes

Contexte général

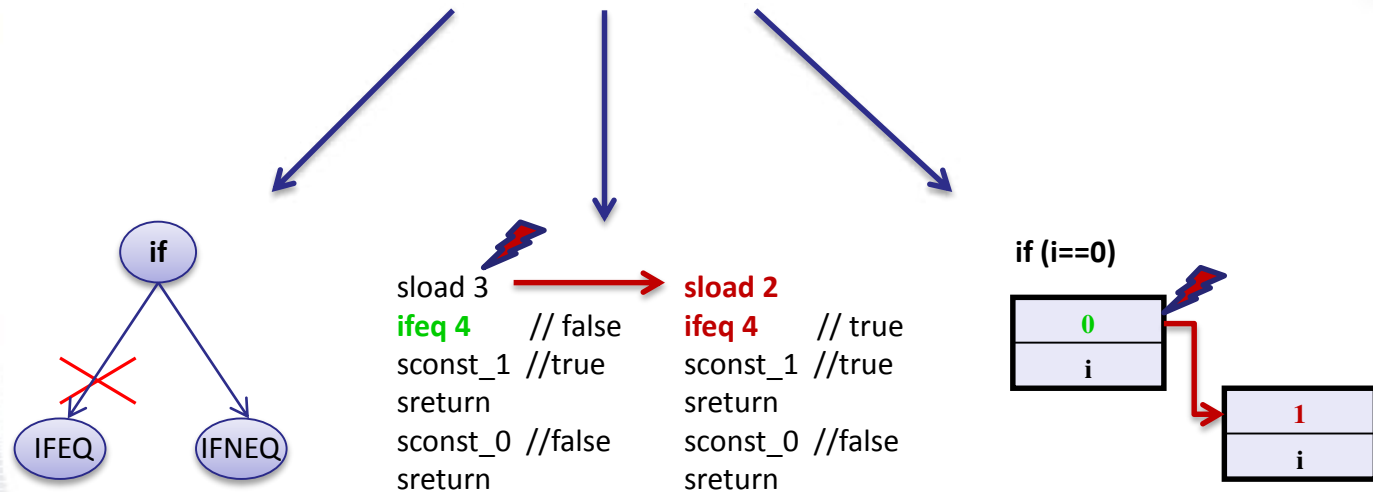
Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

## PERTURB\_CONDITION



# Attaques par injection de fautes

Contexte général

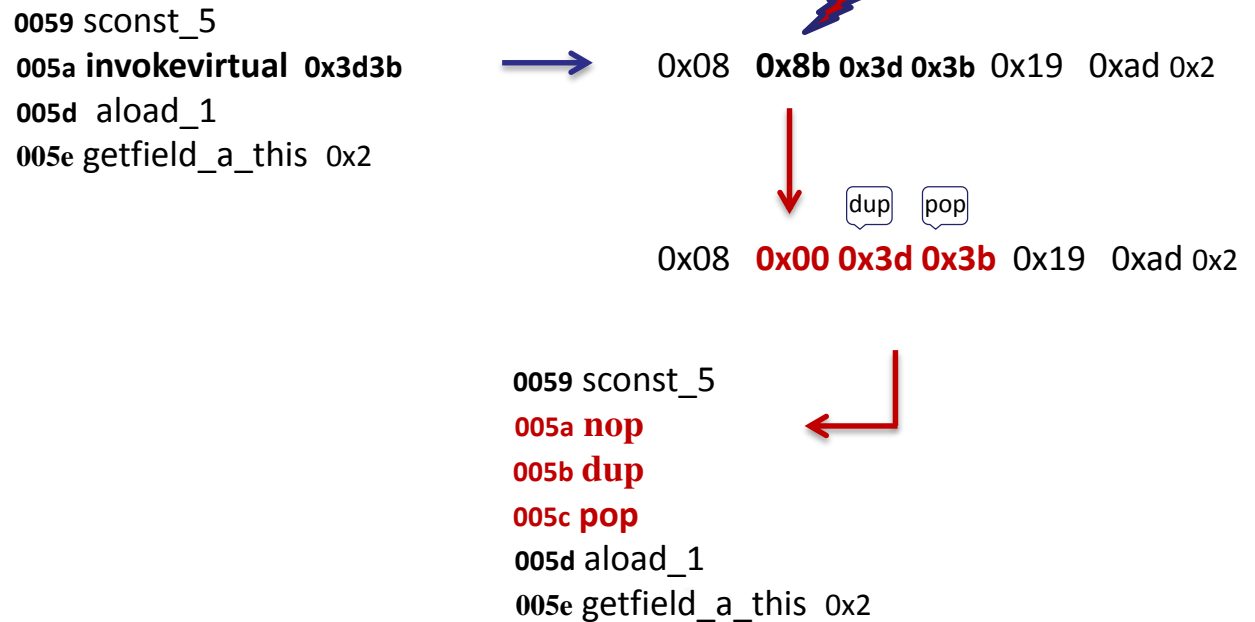
Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

## PERTURB\_INVOKE\_BYPASS



# Contre-mesures applicatives

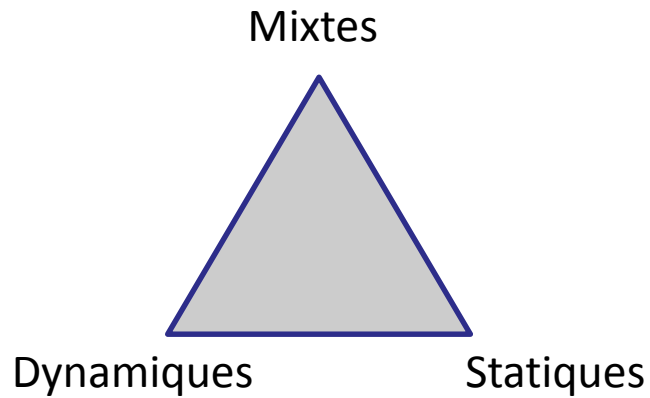
Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion



- Vérification de l'intégrité du code et des données



# Contre-mesures applicatives

Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

- Condition **double if**

```
// condition est un booléen
if (pinIsValidated) {
    if (pinIsValidated) {
        // Opération sensible
    } else {
        /*Attaque détectée! */
    }
} else {
    if (! pinIsValidated) {
        // Accès non autorisé
    } else {
        /*Attaque détectée ! */
    }
}
```

faux



# Contre-mesures applicatives

Contexte général

Problèmes de sécurité

État de l'art


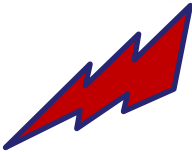
Approche adoptée

Conclusion

- Compteur d'état

```
short compteur_etat=0;
if (compteur_etat ==0) {
    // Opération sensible
    compteur_etat ++;
} else {
    /* Attaque détectée! */
}
/* ... */
if (compteur_etat ==1) {
    // Opération sensible2
    compteur_etat ++;
} else {
    /* Attaque détectée! */
}
```

goto



# Contre-mesures applicatives

Contexte général

Problèmes de sécurité

État de l'art

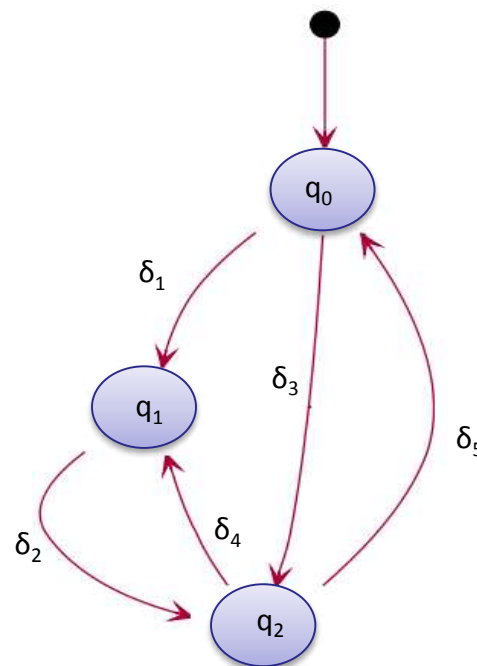
Approche adoptée

Conclusion

- Manuelle, coûteux, fastidieux
- Intégrité du CFG non garantie
- Redondance complète de flux de contrôle ne peut pas être obtenu

# Automate de sécurité

- Triplet de Schneider ( $Q, q_0, \delta$  )
- L'automate est implémenté directement dans le code binaire



Automate de sécurité

Etats	$q_0$	$q_1$	$q_2$
$q_0$	-	$\delta_1$	$\delta_3$
$q_1$	-	-	$\delta_2$
$q_2$	$\delta_5$	$\delta_4$	-

Matrice d'états

# Politique de sécurité CFI

(Intégrité de Graphe de Flot de Contrôle)

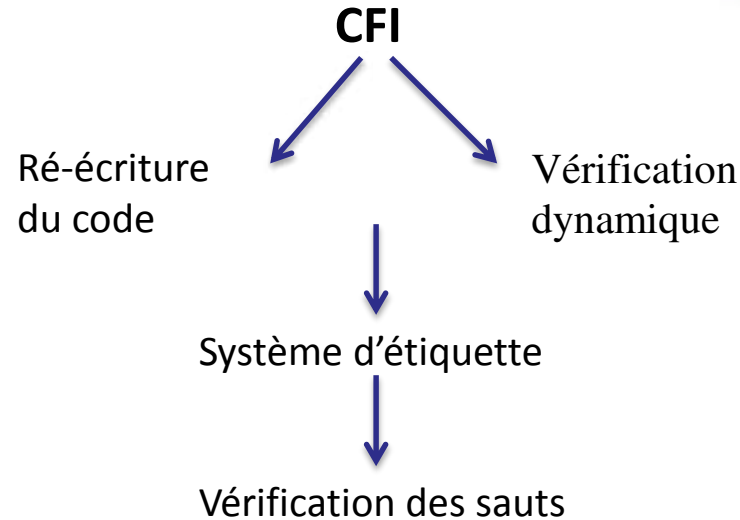
Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion



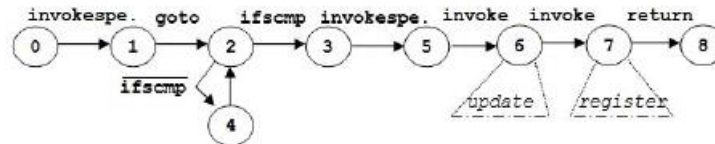
- ✅ Détecter les détournements CFG
- ❌ `PERTURB_CONDITION` non détectée

# Politique de sécurité CFI (Intégrité de Graphe de Flot de Contrôle)

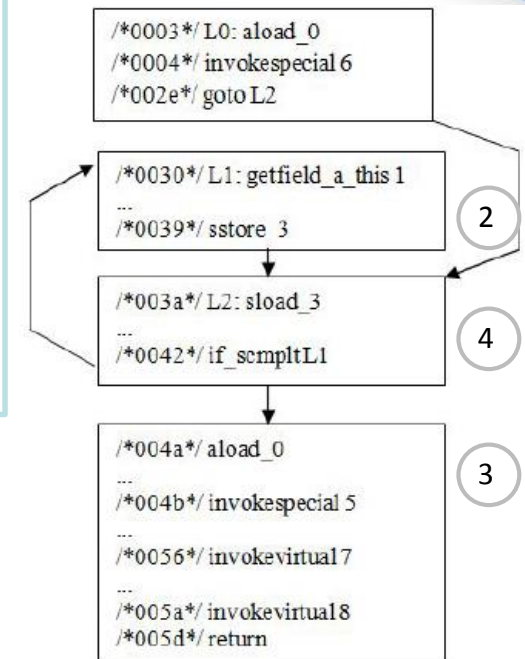
Exemple :

```
protected ProtocolPayment (byte[ ] buffer, short offset ,
byte length ) {
    A[ 0 ] = 0 ; // Initialisation du tableau A
    for (byte j = 0 ; j < buffer [(byte)(offset +12)] ; j++)
        D[ j ] = 0 ; // Initialisation du tableau D

    // Création du PIN
    pin = new OwnerPIN( (byte) NB_ESSAIES_MAX,
    (byte) LONGUEUR_PIN ) ;
    pin . update (myPin, (short ) 0, (byte)
    LONGUEUR_PIN ) ; // Initialisation
    register ( ) ; // Enregistrement de l'instance
}
```



Automate du constructeur de l'applet



CFG du constructeur de l'applet



**PERTURB\_CONDITION non détectée!**

Intégration d'une politique de flot de contrôle  
dans un automate de sécurité

Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

# Implémentation

Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

- Politique CFI
- Surveiller les instructions qui appartient au graphe de flot de contrôle.
- Construire un automate avec une politique de sécurité automatique et statique



# Implémentation

Contexte général

Problèmes de sécurité

État de l'art

**Approche adoptée**

Conclusion

- Un appel/retour Java est coûteux
- Outil d'optimisation en dehors de la carte  
exemple: remplacer un appel de fonction par son code (inlining)
- Adaptation de la VM :
  - construire l'automate de CFG
  - évaluer les instructions modifiant le flot de contrôle



# Implémentation

Contexte général

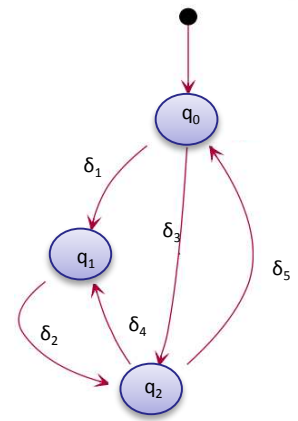
Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion

```
1 int16 BC_if_scmplt ( void) {  
2   if (SM[ frame->currentState][INS]!= vm_pc)  
3     return ACTION_BLOCK;  
4   vm_sp-=2;  
5   if (vm_sp [ 0 ].i < vm_sp [ 1 ] . i )  
6     return BC_goto ( ) ;  
7   if (SM[frame->currentState][NEXT]!=state(vm_pc) )  
8     return ACTION_BLOCK;  
9   vm_pc += 2 ;  
10  return ACTION_NONE;}
```



Etats	q <sub>0</sub>	q <sub>1</sub>	q <sub>2</sub>
q <sub>0</sub>	-	$\delta_1$	$\delta_3$
q <sub>1</sub>	-	-	$\delta_2$
q <sub>2</sub>	$\delta_5$	$\delta_4$	-

Matrice d'états

# Métriques

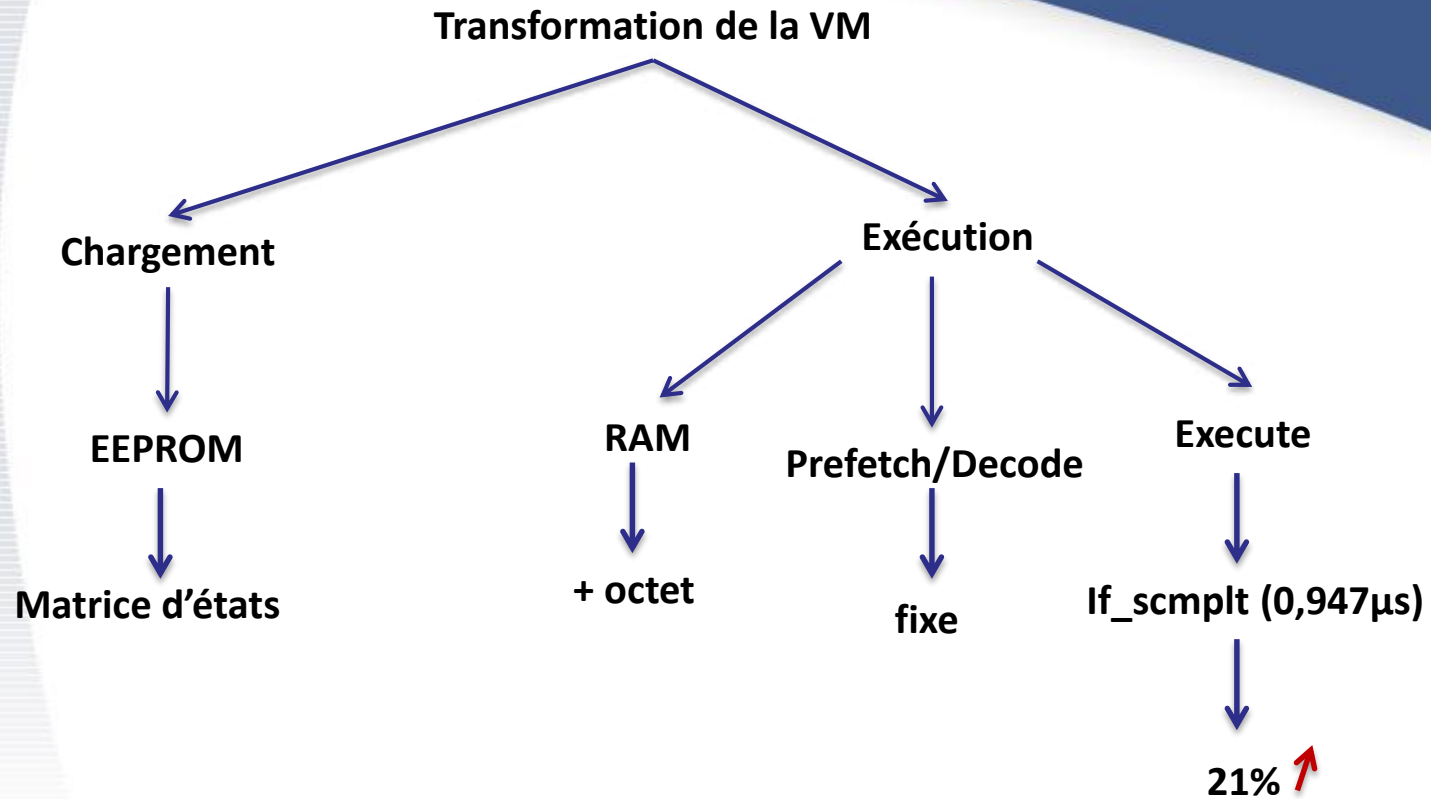
Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion



# Métriques

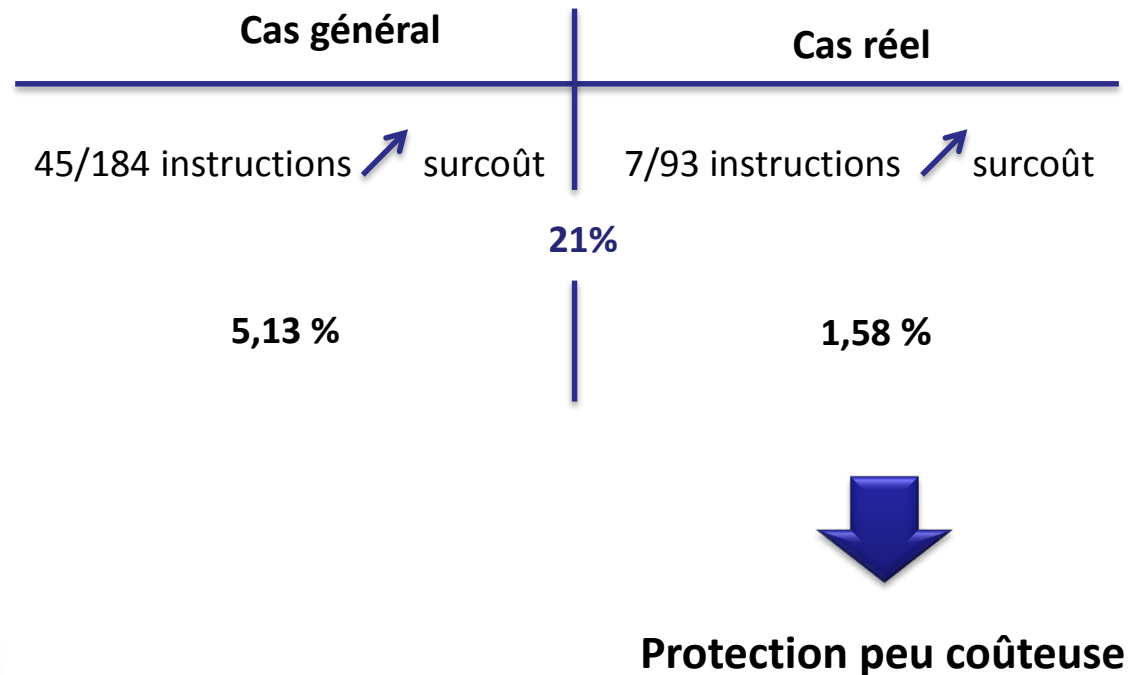
Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

Conclusion



# Conclusion

Contexte général

Problèmes de sécurité

État de l'art

Approche adoptée

**Conclusion**

- Adapter la VM Java Card pour intégrer les fonctions de transition de l'automate
- Construire un automate de sécurité implémentant une politique CFI

*Merci*



**Hanan Tadmori**  
**hanan.tadmori@xlim.fr**

Smart Secure Devices (SSD) Team  
Université de Limoges, 123 Avenue Albert Thomas, 87060 Limoges, France  
[www.secinfo.msi.unilim.fr](http://www.secinfo.msi.unilim.fr)