

Utilisation des méthodes formelles pour l'évaluation et la certification de produits de sécurité

Workshop Certif

Bruno Ezvan

10 octobre 2016

- 1 Introduction
- 2 Présentation de Trusted Labs
- 3 Vérification formelle de circuits numériques pour la certification
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels
- 5 Nouveau cas d'utilisation : la sécurité des mobiles
- 6 Conclusion : objectifs des travaux suivants

Outline

- 1 Introduction
- 2 Présentation de Trusted Labs
- 3 Vérification formelle de circuits numériques pour la certification
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels
- 5 Nouveau cas d'utilisation : la sécurité des mobiles
- 6 Conclusion : objectifs des travaux suivants

Introduction

- Thèse CIFRE à TLabs et au LIP6 (équipe ALSOC) depuis avril 2015
- Sujet : "Vérification par composition pour un haut niveau d'assurance des objets connectés"
- Stage en 2014 chez TLabs sur la vérification formelle de circuits numériques décrits en Verilog.

Outline

- 1 Introduction
- 2 **Présentation de Trusted Labs**
- 3 Vérification formelle de circuits numériques pour la certification
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels
- 5 Nouveau cas d'utilisation : la sécurité des mobiles
- 6 Conclusion : objectifs des travaux suivants

A Global Security Expert

SECURITY CONSULTING



Define and strengthen security tailored to your needs

- Security analysis
- Security certification support
- Security by design
- Certification-ready development
- Tools & training

SECURITY EVALUATION



Get your solution certified by an ITSEF laboratory

- ITSEF laboratory
- Security evaluation facilities
- Accreditations
 - AFSCM
 - CSPN (French light CC)
 - Mobile payment scheme (BCMC)

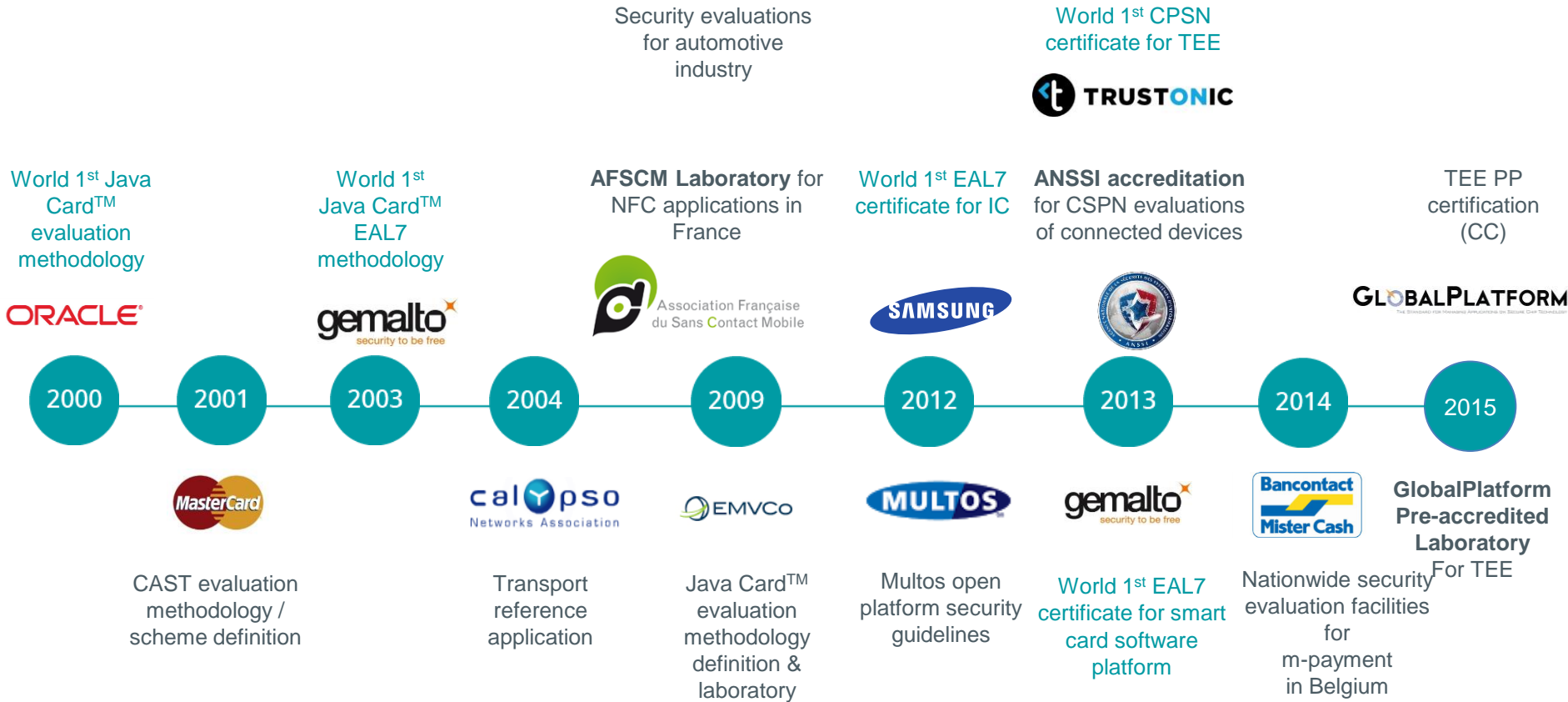
CERTIFICATION SCHEME DEFINITION



Define tailor-made methodologies & certification processes for your industry or ecosystem

- Security requirements
- Evaluation methodologies
- Protection Profiles

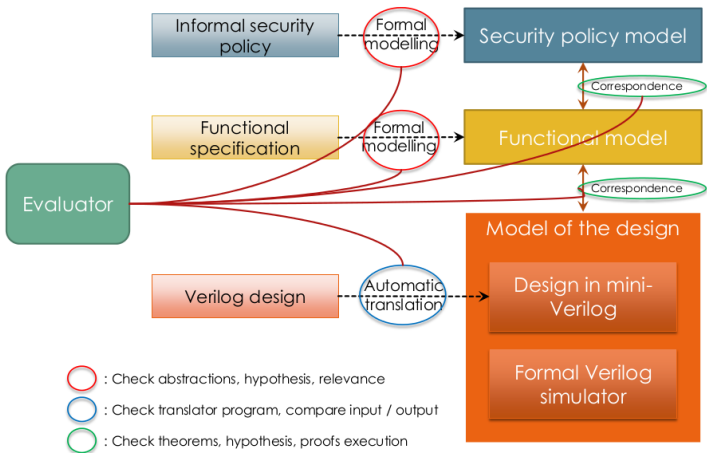
Our Success Timeline



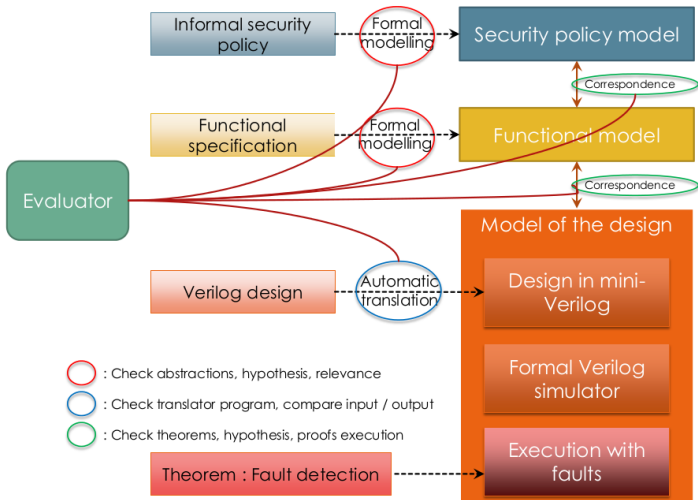
Outline

- 1 Introduction
- 2 Présentation de Trusted Labs
- 3 Vérification formelle de circuits numériques pour la certification**
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels
- 5 Nouveau cas d'utilisation : la sécurité des mobiles
- 6 Conclusion : objectifs des travaux suivants

Vérification formelle de circuits numériques pour la certification



Analyse de robustesse



Modèle du système

Design in mini-Verilog

Formal Verilog simulator

Execution with faults

Model of the design

```
Definition async_process_1 :=  
Always ( Trigger (nil) ;;  
a0 === (Slice (31 - 7) (31 - 0) (Var col_in)) ;;  
a1 === (Slice (31 - 15) (31 - 8) (Var col_in)) ;;  
a2 === (Slice (31 - 23) (31 - 16) (Var col_in)) ;;  
a3 === (Slice (31 - 31) (31 - 24) (Var col_in)) ;;  
r0 === (Xor (Xor (Xor (Net r0_2) (Net r1_3)) (Var a2)) (Var a3)) ;;  
r1 === (Xor (Xor (Xor (Net r1_2) (Net r2_3)) (Var a0)) (Var a3)) ;;  
r2 === (Xor (Xor (Xor (Net r2_2) (Net r3_3)) (Var a0)) (Var a1)) ;;  
r3 === (Xor (Xor (Xor (Net r0_3) (Net r3_2)) (Var a1)) (Var a2)) ;;  
EndBlock ).
```

```
(* Execute a block of statements on a simulation_state q *)  
Definition generic_run_block (b: block) (q: simulation_state) :  
simulation_state :=  
  match b with | EndBlock => q  
               | Seq s b => match s with  
  ..  
               | NullStmt => add_active_event (Evaluate (Block b)) q  
end  
end.
```

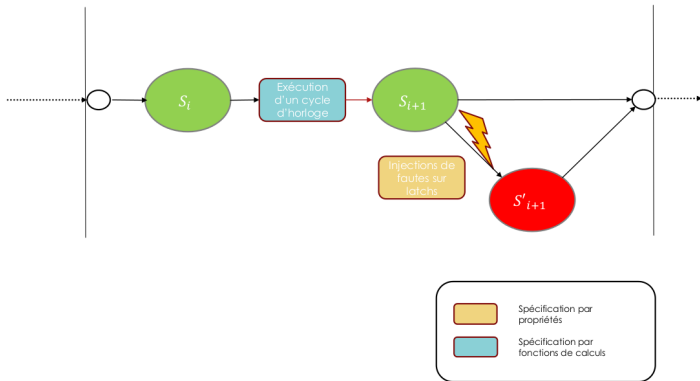
```
Inductive faults_n_cycles (begin end : state) : list state -> nat -> Prop  
:=  
| InjectFaultsOnCycle : ...  
| NoFaultOnCycle : ...  
| NoCycle : ...
```

Modèle de fautes

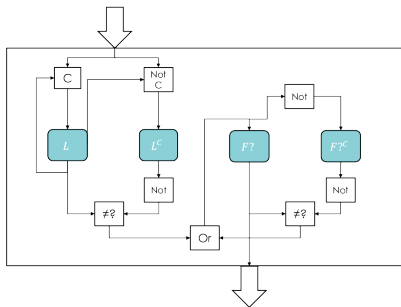
Modèle de fautes

- ❑ Fautes transitoires sur une multitudes de bits des éléments mémorisants
- ❑ Plusieurs localisations de fautes possibles à un instant donné
- ❑ Injections sur plusieurs cycles
- ❑ Hypothèse : impossible d'injecter des fautes sur les latches de la contre-mesure

Formalisation des fautes



Circuit et contre-mesure



Conclusion

- ❑ Difficulté à calquer un modèle de fautes au niveau gates à un niveau d'abstraction différent (RTL)
- ❑ Difficile de formaliser un modèle de faute réaliste

Vérification fonctionnelle d'un circuit

Chiffres

- Vérification fonctionnelle d'un circuit (modulaire) AES (chiffrement symétrique)
 - 38 modules
 - 3500 ligne de Verilog
 - Spécification d'AES : 660 lignes de formalisation en Coq
 - 25 personne-jours pour la preuve

Conclusion

- Modularité : permet d'appliquer l'analyse sur des circuits plus importants
- Méthodes déductives requièrent expertises et travail manuel

Outline

- 1 Introduction
- 2 Présentation de Trusted Labs
- 3 Vérification formelle de circuits numériques pour la certification
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels**
- 5 Nouveau cas d'utilisation : la sécurité des mobiles
- 6 Conclusion : objectifs des travaux suivants

Utilisation méthodes formelles pour l'évaluation sécuritaire de logiciels

Projet FP7 STANCE

Objectif : Mise en place d'une boîte à outil capable de vérifier des propriétés de sécurité d'applications écrites en C, C++ ou en Java.

Trusted Labs

- Evaluer l'utilisation de méthodes formelles d'analyse statique de logiciels dans le cadre de l'évaluation sécuritaire à un niveau d'assurance modéré (non formel)
- Outils : Plateforme Frama-C : plug-in d'interprétation abstraite (Value) et plug-ins basés sur Value
- Cas : ClamAV : un antivirus open source.
 - De nombreux parseurs de données malveillantes : packers, exécutables, images, documents, scripts ...
 - La fonction de sécurité peut être contournée par déni de service
 - L'antivirus peut être un point d'entrée : Remote Code Execution (RCE)

Value : Interprétation abstraite



- Méthode d'approximation des états atteignables d'un programme en utilisant une sémantique abstraite du langage de programmation.
- Permet de déterminer un sur-ensemble des valeurs possibles d'une variable en chaque point du programme

```
int main(void)
{
  int __retres;
  int array[15];
  int i;
  i = 0;
  while (i < 15) {
    array[i] = i;
    i ++;
  }
  __retres = array[12];
  return __retres;
}
```

```
main.c
1 int main() {
2
3   int array[15];
4   int i;
5
6   for (i = 0; i < 15; i++) {
7     array[i] = i;
8   }
9
10  return array[12];
11 }
12
```

Information Messages (0) Console Properties Values WP Goals

Multiple selections

Selection  

Callstack array

[0..14] ∈ [0..14] or UNINITIALIZED

Retours - 1

- Difficulté de mise en place de l'analyse :
 - Non support de directives du compilateur par le parseur de Frama-C
 - Modification des commandes du build de ClamAV pour inclure la libc Frama-C
 - ▶ Quelques inconsistances
- Value ne supporte pas la récursion
 - Pour chaque appel récursif, on peut soit l'ignorer ou utiliser une spécification à la place
 - ▶ En ignorant l'appel, l'analyse n'est plus **sound** (la sur-approximation n'est plus correcte)
 - ▶ La spécification doit être correcte et va généralement ajouter de l'approximation
- Le domaine d'interprétation n'est pas relationnel
 - Trop de problèmes possibles (alarmes) relevés par l'outil
- Impossible d'analyser l'ensemble du logiciel dans le temps impartis : résultats trop imprécis

Retours - 2

Analyse hors contexte de fonctions

Fonction	Nombre de lignes	Ratio lignes/alarmes
scanpe	4830	4
ununpack	815	2,3

- ☐ Nombre d'alarmes trop élevé : trop d'approximations

Limitation des cas d'applications

- ☐ Pas de support de logiciels mixtes assembleurs et C
- ☐ Uniquement safety, pas d'outils clef en main (à notre connaissance) pour analyser les aspects suivant requis par les critères communs :
 - Politique de contrôle d'accès
 - Contrôle du flot d'informations

Conclusion

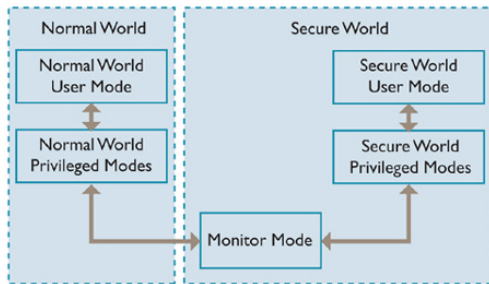
- ☐ L'utilisation de Value et des plugins associés ne se justifie pas dans le cadre d'évaluation sécuritaire à un niveau modéré d'assurance (non formel, EAL1-5) où seuls quelques jours sont attribués à l'analyse du code.

Outline

- 1 Introduction
- 2 Présentation de Trusted Labs
- 3 Vérification formelle de circuits numériques pour la certification
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels
- 5 Nouveau cas d'utilisation : la sécurité des mobiles**
- 6 Conclusion : objectifs des travaux suivants

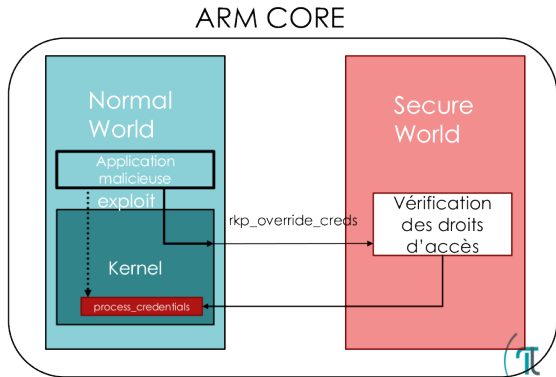
TrustZone

- Extensions de sécurité sur les architectures ARMv7 ou 8
- Le temps d'exécution d'un core se partage entre deux processeurs virtuels
- Isoler les fonctions de sécurité en dehors du "Rich OS" (exemple : Android) dans le Trusted OS
- Normal World → Android
- Secure World → Trusted OS



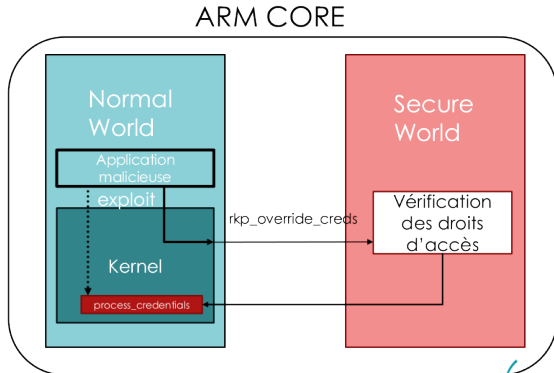
KnoxOut : Description vulnérabilité

- RKP (Real-time Kernel Protection). But : défendre le kernel (Android, Normal OS) d'un potentiel exploit (mitigation)
- Le kernel a une vulnérabilité de type write-what-where
- L'exploit normal consiste à exploiter la vulnérabilité pour écrire la valeur root dans `process_credentials`
- Avec Knox, `process_credentials` est read-only pour le Normal World : l'exploit ne fonctionne pas !



KnoxOut : Description vulnérabilité

- Le Kernel peut demander à RKP le changement des crédits via `rkp_override_creds` (contrôle d'accès)
- RKP vérifie lui même si l'action est autorisée.
- Vulnérabilité : L'action est autorisée si PID=0 hors le pid est contrôlable via le kernel et par la vulnérabilité write-what-where du kernel.



KnoxOut : Conclusion

- La prise en compte des contrôles d'accès lors des évaluations sécuritaires est aussi importante que la memory-safety.
- Le contrôle du flot d'informations l'est aussi dans le cadre de produits de sécurité fournissant des mécanismes d'isolation

Outline

- 1 Introduction
- 2 Présentation de Trusted Labs
- 3 Vérification formelle de circuits numériques pour la certification
- 4 Utilisation de méthodes formelles pour l'évaluation sécuritaire de logiciels
- 5 Nouveau cas d'utilisation : la sécurité des mobiles
- 6 Conclusion : objectifs des travaux suivants

Conclusion : objectifs des travaux suivants

Objectifs

- Mettre en place un outillage facilitant l'évaluation sécuritaire de système de l'internet des objets
- Automatisation
- Focus sur propriété sécuritaire ou fonctions de sécurité moins ciblées par d'autres techniques
 - Contrôle du flot d'information
 - Contrôle d'accès

Moyens

- Analyse au niveau assembleur
 - Éviter le problème l'écart entre les différents niveau d'abstraction (exemple des fautes)
 - Éviter de limiter l'analyse à un sous-ensemble particulier dans langage haut niveau (exemple : non support d'extensions du compilateur)
 - Inclure les spécificités de l'architecture matérielle (exemple : TrustZone)
- Analyse compositionnelle

Questions ?

Questions ?