Vérification formelle de la bibliothèque USBctrl de WooKey à l'aide de FRAMA-C

Cyril DEBERGE

Stage réalisé au sein de l'ANSSI (SDE/DST/LSL)

Encadré par Patricia MOUY, SGDSN/ANSSI/SDE/DST/LSL et Philippe THIERRY, SGDSN/ANSSI/SDE/DST/LAM

Plan de la présentation

Plan

Objectifs du stage

Vérification formelle de la bibliothèque USBctrl de WooKey :

- Preuve de l'absence de run time errors (RTE)
- Preuve fonctionnelle de la bibliothèque

Objectifs :

- Garantie des fonctions de sécurité assurées par la bibliothèque USBctrl :
  - Confidentialité
  - Intégrité
  - Disponiblité
- Développer une méthodologie d'analyse avec Frama-C, reproduisible sur d'autres codes

Plan

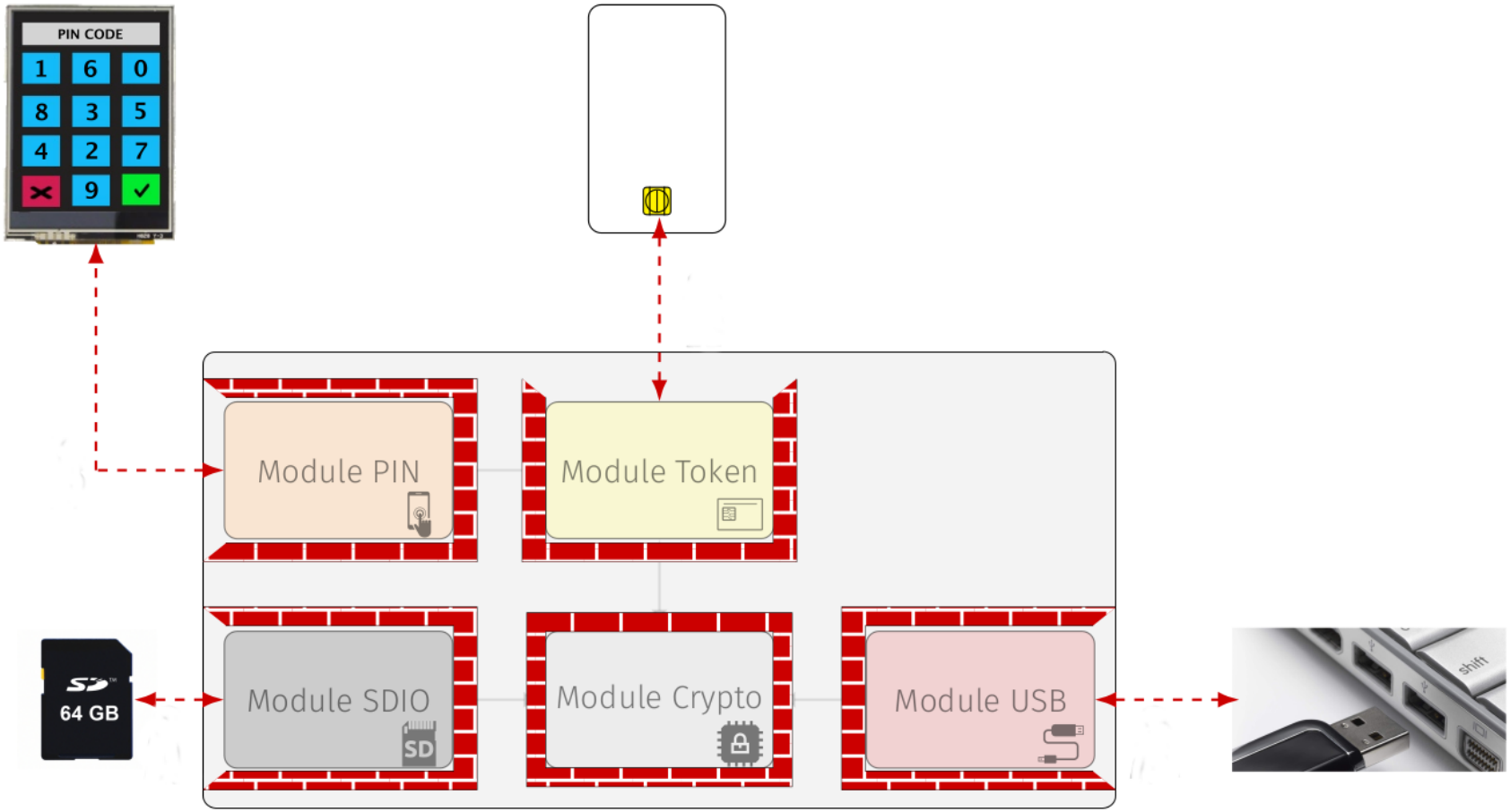
Plan

Projet WooKey

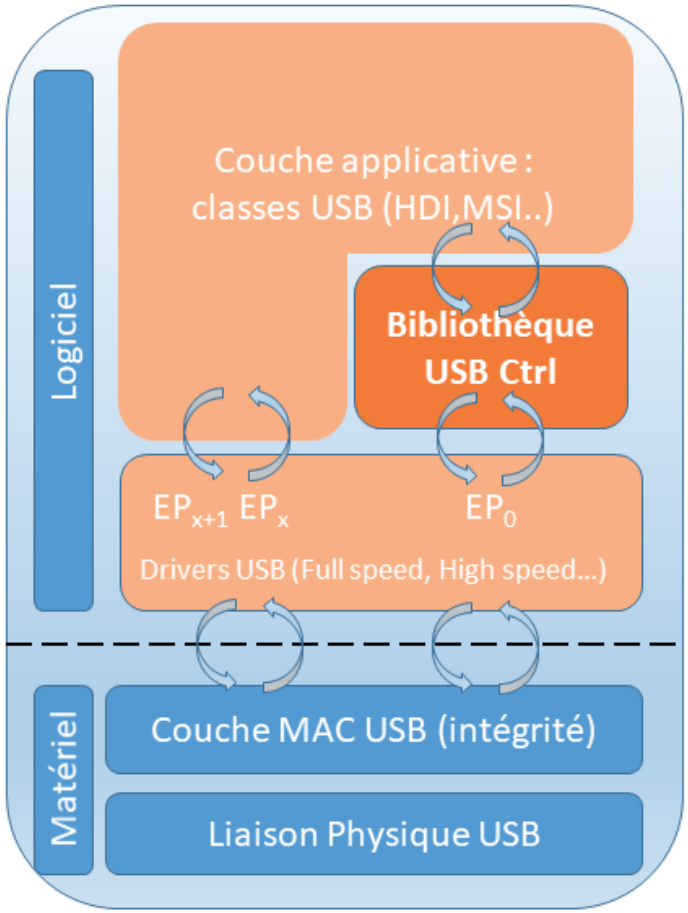
Projet Wookey :

- Clé USB chiffrante sécurisée
- Projet open source et open Hardware
- Nombreuses propriétés de sécurité :
  - Défense en profondeur
  - Architecture modulaire (cloisonement des différents modules)
  - Mise à jour logicielle sécurisée
  - Utilisation de la cryptographie
  - Deux facteurs d'authentification

Architecture de WooKey

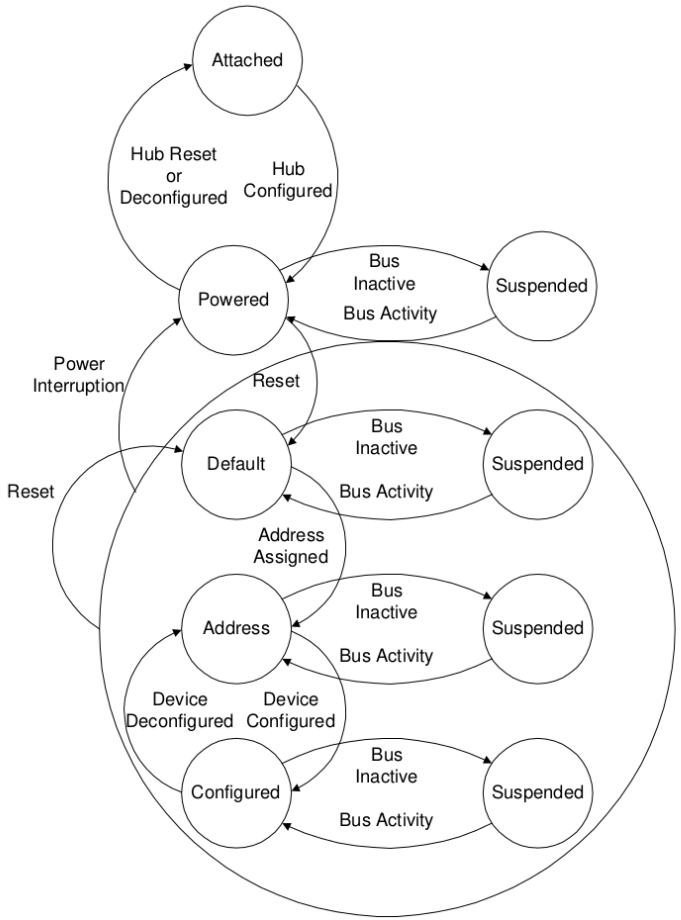


Plan



- Bibliothèque écrite en langage C
- Responsable du contrôle de la couche USB 2.0, en espace utilisateur (réponses aux sollications de l'hôte, gestion de la phase de négociation entre l'hôte et le périphérique)
- Gérer la machine à états de l'USB 2.0, sans nécessiter d'actions complexes de la part des couches plus élevées du périphérique
- Abstraction du driver USB pour permettre une portabilité complète, quelque soit la classe USB du périphérique

Machine à états de l'USB 2.0



Architecture de la bibliothèque :

- 5 fichiers, un peu plus de 2000 lignes de code (standards c90, c99 et c11) :
  - Déclaration du contexte USB, des interfaces, des endpoints
  - Gestion des requètes provenant de l'hôte ou du périphérique
  - Gestion des différents états de l'USB 2.0
- 2 drivers USB actuellement codés dans WooKey : full speed et high speed (interface à travers des alias dans la bibliothèque USBctrl)

Plan

Plan

Définition

Raisonnement rigoureux, à l'aide de logique mathématique, sur des programmes informatiques afin de démontrer leur validité par rapport à une certaine propriété

Objectif

Avoir la preuve que, quelle que soit l'entrée fournie au programme, si elle respecte une propriété donnée, alors le programme fera ce qui est attendu

Notions importantes :

- Complétude : Absence de faux positif (sous approximation)
- Correction : Absence de faux négatif (sur approximation)

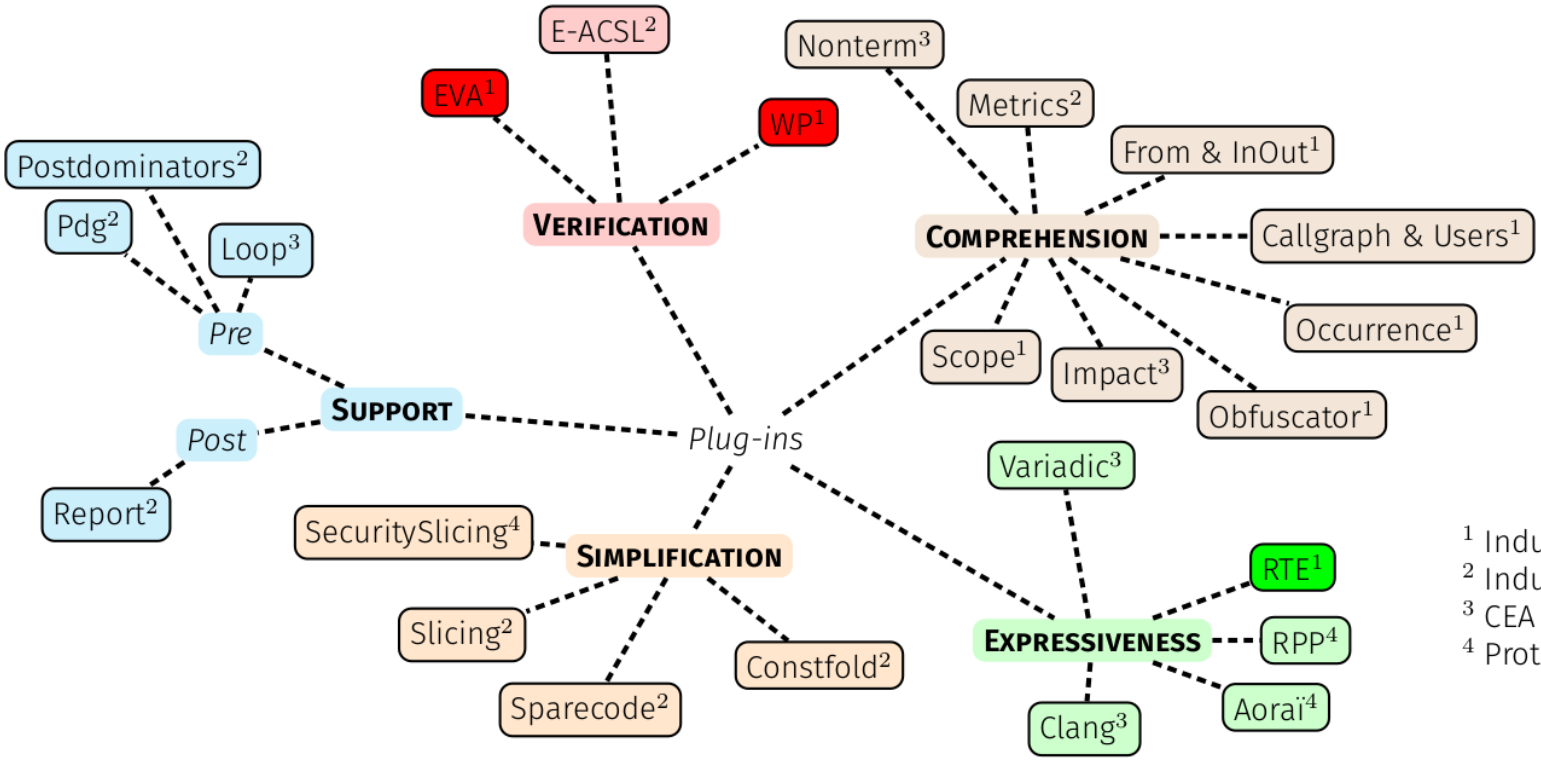
Plan

Présentation de Frama-C

Frama-C (Framework for modular analysis of C programs) :

- Plate-forme open-source, modulaire et collaborative dédiée à l'analyse du langage C
- Plate-forme développée par le CEA-LIST et l'INRIA
- Analyse statique et dynamique
- Utilisation en ligne de commandes ou à l'aide d'une interface graphique
- Complétude (analyse dynamique) et / ou correction (Greffons EVA, WP...)

Greffons open-source du CEA

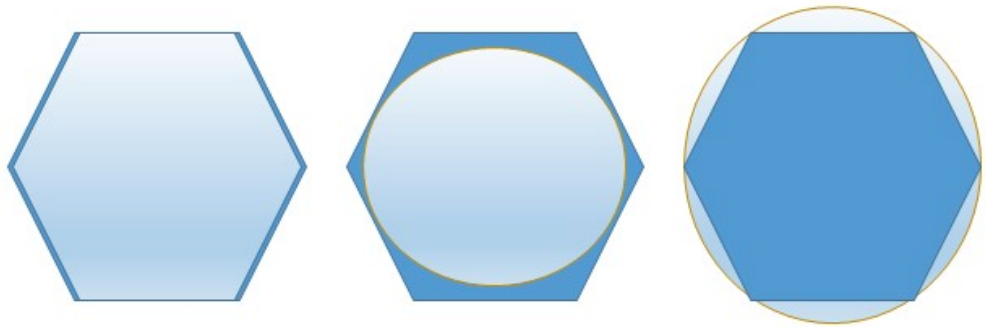


Greffon EVA

EVA (Evolved Value Analysis) :

- Analyse statique et automatique du code, à l'aide d'une interprétation abstraite
- Estimation des valeurs possibles des différentes variables présentes dans le code
- Détection des RTE potentielles (accès mémoire invalide, variables non initialisées , division par 0...)
- Nécessite un point d'entrée dans le code

Schéma d'analyse :

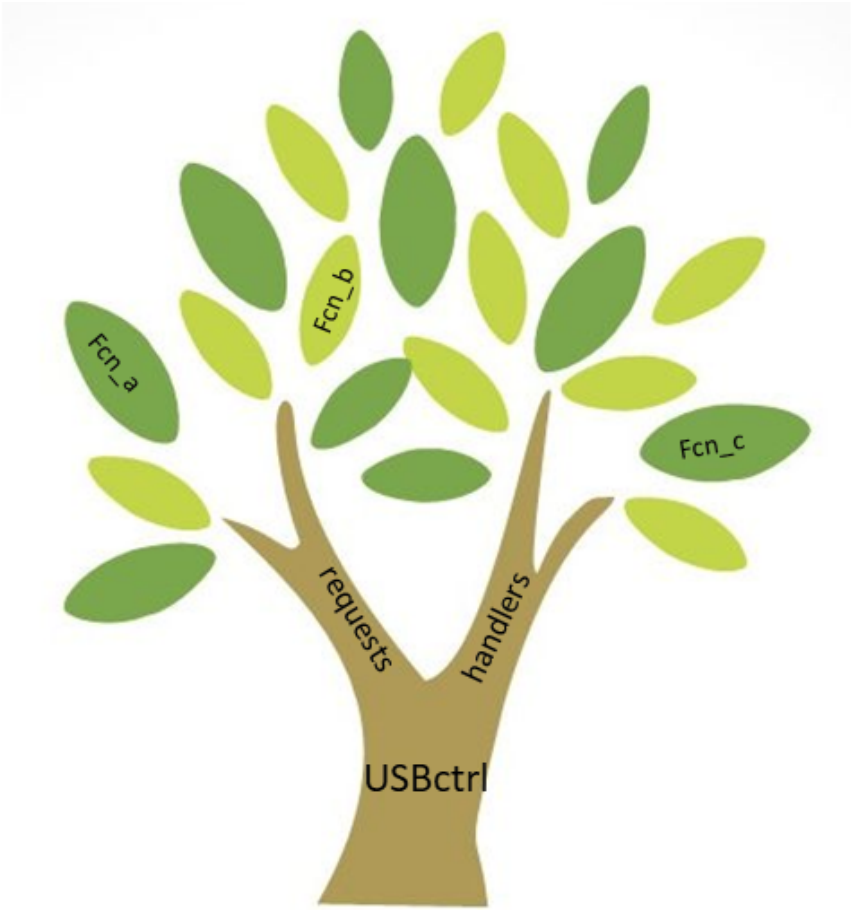


Greffon WP

WP (Weakest Precondition) :

- Preuve mathématique des propriétés fonctionnelles du code analysé (exemple de propriété fonctionnelle : absence de RTE)
- Génère des conditions de vérification vérifiées à l'aide de prouveurs automatiques externes à l'outil Frama-C
- Modulaire : analyse fonction par fonction
- Basé sur la logique de Hoare :
  - $\{Pre\}$ P $\{Post\}$: Si Pre est satisfait et P termine alors Post est satisfait après exécution de P (correction partielle)
  - $[Pre]P[Post]$: Si Pre est satisfait alors P termine et Post est satisfait après exécution (correction totale : objectif de WP)
  - Plus faible précondition Pre : propriété Pre la plus simple qui doit être valide avant P de telle manière que Post est vérifié après P

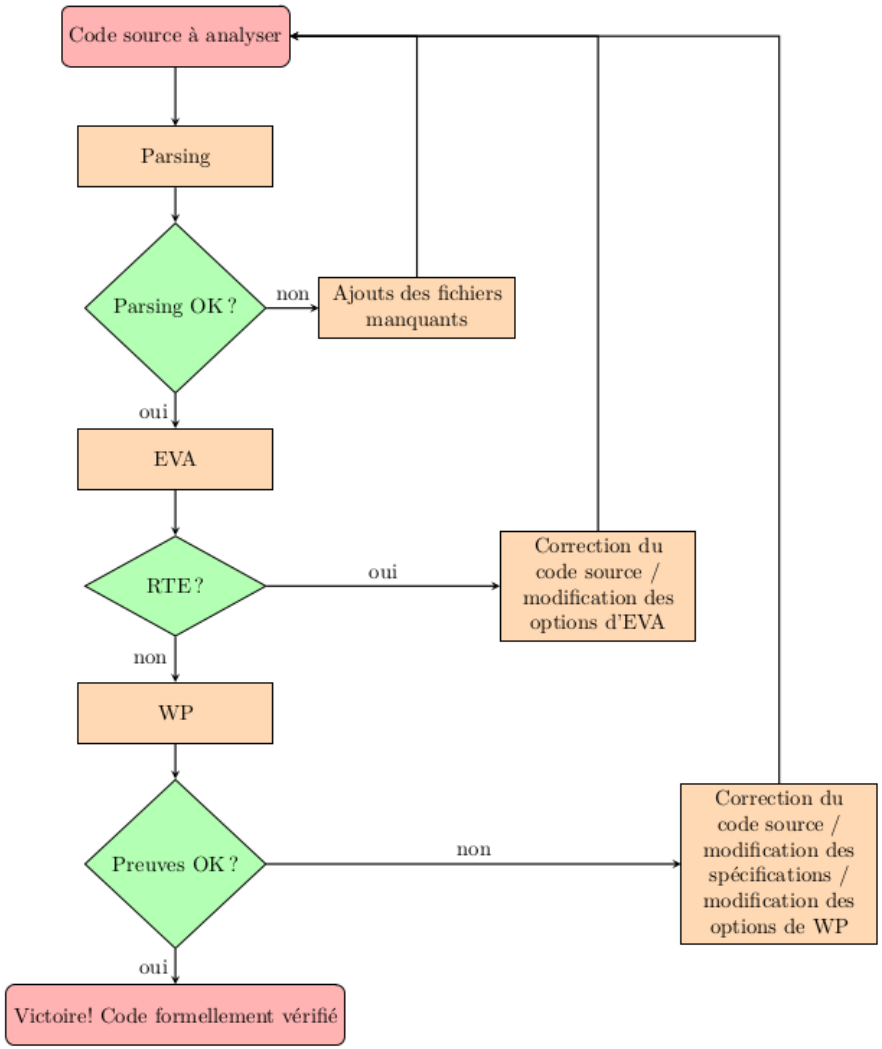
Schéma d'analyse



Analyse modulaire, qui part des fonctions élémentaires (les feuilles) vers les fonctions complexes (les branches)

Plan

Stratégie d'analyse mise en oeuvre



Stratégie d'analyse mise en oeuvre

Avantage d'une telle stratégie :

- L'utilisation d'EVA permet de garantir l'absence de RTE pour des propriétés simples
- WP permet de garantir l'absence de RTE pour des propriétés plus complexes (par exemple, terminaison correcte de la fonction)
- WP utilise les résultats d'EVA pour ses obligations de preuve
- Résultats obtenus plus "forts" qu'avec une utilisation séparée des deux greffons

Contraintes :

- Modifier le moins possible le code de la bibliothèque USBctrl et son architecture (sauf correction du code)
- Utiliser la version open source des greffons (être le plus reproduisible possible)

Plan

Plan

Parsing de la bibliothèque avec Frama-C afin de s'assurer que FramaC dispose de l'ensemble des fichiers pour pouvoir analyser le code

Ajout de la libc de wookey + fichiers du driver + fichiers de configuration de la lib USBctrl : Analyse réalisée dans une configuration particulière

Plan

Analyse avec EVA

L'analyse avec EVA doit tenir compte en particulier de deux paramètres importants :

- La précision de l'analyse : EVA doit disposer de suffisamment de "carburant" pour analyser sans trop d'approximation les boucles et les structures conditionnelles (switch, if/else...)
  - Précision paramétrable à travers les options d'EVA : unroll loop et slevel en particulier. Plus ces paramètres sont élevés, plus la précision est importante, mais plus le temps de calcul augmente
  - Le post-traitement des logs d'EVA et l'utilisation de l'interface graphique permet de vérifier que la précision atteinte est satisfaisante

Analyse avec EVA

L'analyse avec EVA doit tenir compte en particulier de deux paramètres importants :

- Le taux de couverture du code : pour avoir confiance dans les résultats obtenus par EVA, il est nécessaire que tout le code soit analysé
  - Création d'un point d'entrée dans le code pour Frama-C : une fonction main, qui appelle les différentes fonctions de la lib USB après avoir initialisé le contexte d'appel de chaque fonction
  - Nécessité de créer plusieurs contextes d'appels des différentes fonctions de la bibliothèque USB et du driver pour couvrir les cas d'erreur (sinon, en condition nominale, EVA n'y rentre pas). Exemple, appel d'une fonction avec un pointeur null

Analyse avec EVA

Résultats obtenus et travail à faire :

- Taux de couverture quasiment intégral de la lib USB et du driver :
  - Toutefois, tous les embranchements possibles ne sont pas encore analysés par EVA : certains embranchements dépendent de l'état du driver et de la lib (au sein de la machine à état de l'USB 2.0) ou de cas d'erreurs pas encore testés.
- Une dizaine de RTE découverts, dans la bibliothèque USBctrl et dans le driver :
  - Débordements de tableaux
  - Débordements d'entier non signés
  - Division par 0
  - accès mémoire invalide
  - Variables non ititialisées
- RTE patchées, à l'exception d'une RTE dans le driver

Plan

Analyse avec WP

L'analyse du code avec WP nécessite de définir le contrat de l'ensemble des fonctions de la bibliothèque USBctrl ainsi que pour certaines fonctions du driver

```
/*@
   requires INT_MIN < val;
   ensures \result >= 0;
   ensures (val >= 0 ==> \result == val) &&
           (val < 0 ==> \result == -val);
   assigns \nothing ;
*/
int abs(int val){
   if(val < 0) return -val;
   return val;
}
```

Analyse avec WP

Travail réalisé :

- environ 4/5 des fonctions de la lib USBctrl spécifiées totalement : 34 fonctions sur 40
- sans compter la spécification des effets de bord (les adresses mémoires potentiellement modifiées par les fonctions), 38 fonctions sur 40 sont spécifiées
- Pour les fonctions du driver appelées par la bibliothèque USB, il en reste 4 à spécifier totalement, sur 16

Analyse avec WP

Plusieurs problématiques soulevées pour terminer de spécifier la bibliothèque USBctrl :

- Gérer les variables statiques quand elles sont utilisées dans les spécifications des fonctions
- Gérer les variables volatiles
- Niveau de détail des spécifications et portabilité de l'analyse
- Gérer certains cast, qui rendent l'analyse de WP incorrecte

Merci de votre attention.

Questions ?