MINISTÈRE DE LA DÉFENSE

RENCONTRES DGA/INRIA

Sécurité et sûreté des implémentations logicielles et matérielles.

DÉLÉGATION GÉNÉRALE POUR L'ARMEMENT

Plan

- Architecture des produits de sécurité.
- Les limites du cycle actuel de développement.
- Une première approche : l'analyse de flots sécurisée.

Architecture des produits de sécurité

- Les produits de sécurité considérés dans cet exposés sont des produits permettant de communiquer sur un réseau non forcément maîtrisé des informations en garantissant :
  - la confidentialité,
  - l'intégrité.
- Exemples
  - chiffreur IP,
  - téléphone chiffrant,
  - chiffreur d'artère.

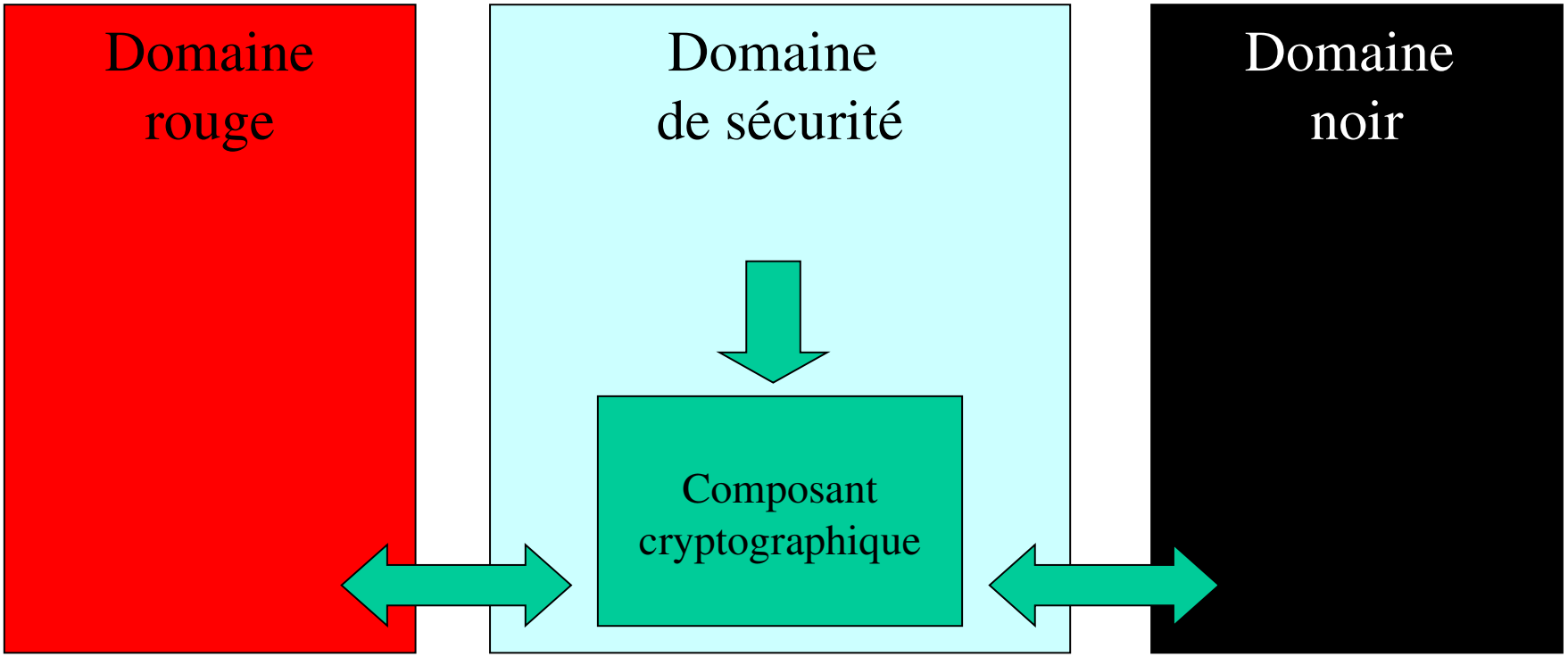
Les mécanismes cryptographiques

- Les informations classifiées peuvent être déclassifiées par l'utilisation de mécanismes (algorithmes, protocoles) ayant reçu un agrément de principe par la DCSSI.
- Ces mécanismes sont en général montrés sûrs par une preuve de sécurité.
- La problématique :

Comment garantir que le produit de sécurité implémente correctement les mécanismes cryptographiques prouvés?

Architecture rouge/noire

La sécurité repose sur le composant cryptographique qui fait l'objet d'un développement soigneux.



Fonctionnalité d'un composant cryptographique

- Boite de chiffrement
  - données sensibles non chiffrées transitant dans le composant.
- Coffre fort :
  - ensemble de biens fini et connu, contenus dans le composant cryptographique permettant la bonne exécution de services,
  - chiffrement,
  - négociation de clé,
  - …

L'attaquant

- A la maîtrise de tout ou partie des interfaces du composant, et est capable d'exécuter n'importe quelle commande sur n'importe quelle interface qu'il contrôle dans n'importe quel ordre.

- Son objectif est de trouver un séquencement de commandes qui lui permettrait de déduire tout ou partie des biens sensibles.

Démarche actuelle de développement

- La méthodologie de développement est le cycle en V
- Le test (test fonctionnel, test d'intégration, évaluation) est la pierre angulaire permettant d'avoir confiance en l'implémentation.

Les limites du cycle de développement en V

- L'attaquant gagne si il trouve un séquencement de commandes compromettant.
- Par contraposée, il est nécessaire d'exécuter exhaustivement toutes les séquences pour que la garantie de défaite de l'attaquant soit totale.
- Le nombre de fonctionnalités proposées par un composant augmente.
- La loi de croissance des tests associés pour un niveau de sécurité constant est supérieure à celle du développement.

Comparaison avec le logiciel « critique»

- Les garanties recherchées pour les produits de sécurité sont légèrement différentes de celles recherchées dans l'élaboration de logiciel critique :
  - un produit de sécurité peut arrêter son exécution, il n'a pas vocation à exécuter son service quoi qu'il arrive (détection d'attaques) ;
  - un logiciel critique peut omettre certains enchainements improbables.

Une première approche

- Un projet de recherche est en cours en collaboration avec le LSV de l'ENS Cachan.
- La démarche proposée est l'analyse de flots sécurisés.

L'analyse de flots sécurisés

- L'analyse de flots sécurisés se déroule en 3 étapes essentielles :
  - étude de la grammaire du langage de description,
  - typage des informations manipulées dans un programme,
  - exécution d'un algorithme de vérification de la satisfiabilité des contraintes.

Grammaire du langage de description

- On considère un ensemble de types x ordonné et un prédicat sur les variables d'un programme.
- La grammaire du langage est analysée exhaustivement.
- Un ensemble de règles de typage traduisant la politique de sécurité est déduite de la sémantique du langage.
- Si un programme est correctement typé, alors la politique de sécurité est respectée.

Le typage des informations

- Chaque information manipulée aux entrées et sorties du composant est typée.
  - Exemple « rouge », « noir », …
- Une relation d'ordre (qui peut être partielle) est définie entre les types
  - Exemple : rouge > noir.

Algorithme de vérification automatique de satisfiabilité des contraintes

- Entrées :
  - description du composant,
  - type des entrées,
  - règles de déduction de types.
- Sortie :
  - un typage correct du programme a été trouvé,
  - $\perp$ a été démontré.

La garantie apportée

- Ce que cette approche nous fait gagner
  - la garantie du cloisonnement entre niveaux jusqu'aux entrées sorties des sous blocs (ou sous fonctions) implémentant les mécanismes prouvés,
  - l'impossibilité pour l'attaquant de by-passer ces implémentations.
- Ce qu'il reste à réaliser
  - la preuve d'équivalence entre les mécanismes et leur implémentation.