Analyse statique pour la sécurité chez Airbus

état des lieux et nouveaux besoins

Sarah Zennou

31 janvier 2017

Bio

- 2004 doctorat en informatique sur le model-checking de systèmes concurrents
- 2005 post-doc à l'X sur l'analyse statique pour la sûreté
- 2005-2008 ingénieur-chercheur au CEA LIST en analyse statique pour la sûreté
- 2008-2010 ingénieur de recherche chez Airbus en analyse statique pour la sûreté
- 2010- ingénieur de recherche chez Airbus en analyse statique pour la sécurité

Rôle: expert en analyse statique pour la sécurité

2 types de missions

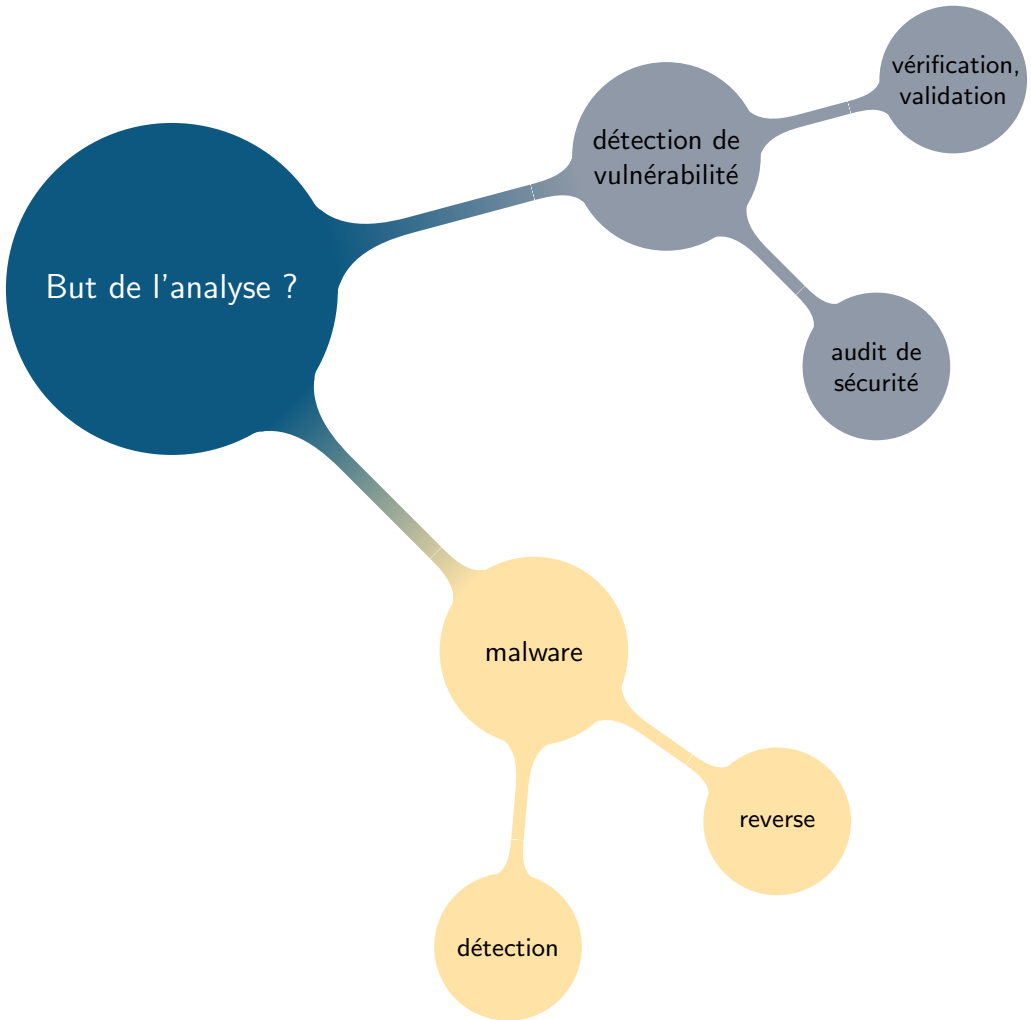
- veille technologique: conseil aux opérationnels pour le choix de technologies existantes
- R&T: propositions de solutions innovantes pour les nouvelles problématiques

Avant-propos

Cet exposé n'est pas

- un catalogue des offres de services en matière de cybersécurité pour les entreprises extérieures
- exhaustif: il peut exister des initiatives/travaux dans le groupe dont je n'ai pas connaissance

Contexte de l'analyse statique automatique pour la sécurité



Part I

Audit de code

Vérification pour la sécurité *vs* audit de sécurité

Vérification

- le but est de construire des systèmes plus sûrs, plus sécurisés
- c'est une utilisation défensive des méthodes formelles

Vérification pour la sécurité *vs* audit de sécurité

Vérification

- le but est de construire des systèmes plus sûrs, plus sécurisés
- c'est une utilisation défensive des méthodes formelles

mais on a du mal à spécifier formellement ce qu'est une propriété de sécurité

Vérification pour la sécurité *vs* audit de sécurité

Vérification

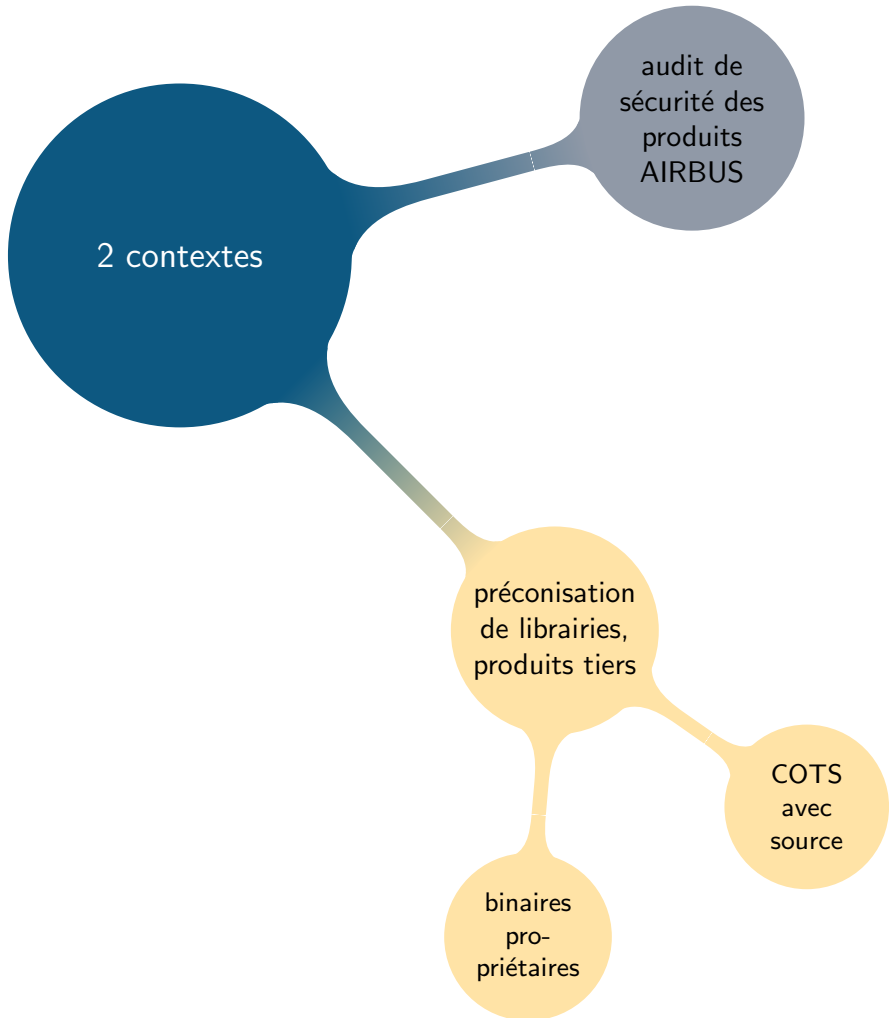
- le but est de construire des systèmes plus sûrs, plus sécurisés
- c'est une utilisation défensive des méthodes formelles

mais on a du mal à spécifier formellement ce qu'est une propriété de sécurité

Audit

- le but est de trouver une surface d'attaque d'un système pour le détruire
- utilisation des méthodes formelles de façon offensive
- ne sera jamais un substitut à la vérification

Contexte de l'audit de code



Scénarios typiques d'audit 1/2

Constat

- des données mal formatées peuvent provoquer des erreurs à l'exécution
- toute donnée venant de l'extérieur doit être validée avant d'être utilisée

Exemples de scénarios d'audit

- Scenario 1:
  1. choisir une entrée manipulable par l'attaquant
  2. chercher si elle a un impact sur un élément sensible/vulnérable
     $\Rightarrow$ data tainting, slicing en avant
  3. vérifier si/comment c'est exploitable, i.e. est-ce qu'on peut manipuler la donnée de façon à altérer le comportement du programme ?
     $\Rightarrow$ slicing en arrière

Scénarios typiques d'audit 2/2

- Scenario 2:
  1. choisir un point sensible dans le code (ex: fonction critique nécessitant de forts privilèges)
  2. vérifier si ce point est accessible depuis une entrée contrôlable

     $\Rightarrow$ slicing en arrière

Retour d'expérience de l'utilisation des méthodes formelles pour la détection de vulnérabilité

Dans le cadre de la vérification

- peu d'outils *sound* disponibles qui ne traitent pas que la *safety*
- mais thème de recherche très actif au niveau source et binaire

Dans le cadre de l'audit

- retours très mitigés du fait de
  - du temps d'analyse limité donc
    - tout ne peut pas être analysé au même niveau de précision
      $\Rightarrow$ possibilité de traiter le code par "bouts", couper certains pans de l'analyse, etc.
    - pas le temps de regarder tous les faux positifs
      $\Rightarrow$ il faut donner des indices/conseils aux utilisateurs pour les guider dans la résolution
    - fort besoin d'analyse arrière
      $\Rightarrow$ point faible des analyseurs

Retour d'expérience de l'utilisation des méthodes formelles pour la détection de vulnérabilité (suite)

Dans le cadre de l'audit

- la diversité des languages présents dans un audit: *toolchain*, Makefile, C++ pas/peu géré, mélange de source et binaire, présence de scripts, etc.
- il y aura toujours des nouvelles propriétés non modélisées dans l'outil

  $\Rightarrow$ simplifier les interactions avec l'analyste. Elles doivent être dans la sémantique du programme à analyser et pas dans la théorie sous-jacente

Part II

Analyse de malware

Problématique spécifique au malware

Contexte

- mêmes problèmes que l'analyse de binaire propriétaire car pas de connaissance a priori du comportement

Problématique spécifique au malware

Contexte

- mêmes problèmes que l'analyse de binaire propriétaire car pas de connaissance a priori du comportement
- le nombre de malware que Airbus reçoit chaque mois ne fait que croître

Problématique spécifique au malware

Contexte

- mêmes problèmes que l'analyse de binaire propriétaire car pas de connaissance a priori du comportement
- le nombre de malware que Airbus reçoit chaque mois ne fait que croître
- le nombre d'employés capables de les analyser ne croît pas dans les mêmes proportions

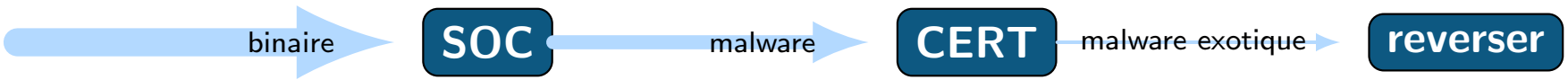
Problématique spécifique au malware

Contexte

- mêmes problèmes que l'analyse de binaire propriétaire car pas de connaissance a priori du comportement
- le nombre de malware que Airbus reçoit chaque mois ne fait que croître
- le nombre d'employés capables de les analyser ne croît pas dans les mêmes proportions

$\Rightarrow$ fort besoin d'automatiser leur traitement

Schema typique de traitement d'un malware



Problématiques

1. distinguer les malwares spécifiques des malwares opportunistes (phishing, etc.), ces derniers étant largement majoritaires
2. mettre des priorités sur ce qui doit être analysé manuellement

Considérations concernant la détection et la priorité de traitement des malware

- concernant la détection
  - détection $\neq$ classification
  - approche statique préférée:
    - pour passage à l'échelle (sinon autant de VM en parallèle que de malware à analyser)
    - moins dangereuse que l'exécution sandboxée de malware
- concernant les priorités
  - moins de volume que pour la détection donc plus de place aux méthodes sémantiques
  - le CFG c'est trop bas niveau
  - quelques pistes d'ordonnancement:
    - les plus exotiques en premier
    - les plus sophistiqués en premier (lien avec l'obfuscation, packing)
    - les plus dangereux en premier (en fonction des ressources attaquées, etc.)

Considérations concernant la détection et la priorité de traitement des malware

- concernant la détection
  - détection $\neq$ classification
  - approche statique préférée:
    - pour passage à l'échelle (sinon autant de VM en parallèle que de malware à analyser)
    - moins dangereuse que l'exécution sandboxée de malware
- concernant les priorités
  - moins de volume que pour la détection donc plus de place aux méthodes sémantiques
  - le CFG c'est trop bas niveau
  - quelques pistes d'ordonnancement:
    - les plus exotiques en premier
    - les plus sophistiqués en premier (lien avec l'obfuscation, packing)
    - les plus dangereux en premier (en fonction des ressources attaquées, etc.)

  analyse automatique statique à gros grain

Conclusion

Conclusion

- convaincus de l'utilité d'outils sémantiques automatiques pour faire de la sécurité
- leur utilisation en pratique reste difficile
- besoins

Conclusion

- convaincus de l'utilité d'outils sémantiques automatiques pour faire de la sécurité
- leur utilisation en pratique reste difficile
- besoins