

Application de la démarche CyberEdu à l'embarqué

Olivier Levillain

Journée CyberEdu à Colmar

Plan

Définitions de l'embarqué

JavaCard

Arduino

Android

Quelques recommandations générales

Qu'est-ce que l'embarqué

Caractérisation de l'embarqué

Qu'est-ce que l'embarqué

Caractérisation de l'embarqué

- ▶ ressources limitées
 - ▶ CPU
 - ▶ RAM
 - ▶ batterie
 - ▶ stockage
 - ▶ connectivité

Qu'est-ce que l'embarqué

Caractérisation de l'embarqué

- ▶ ressources limitées
 - ▶ CPU
 - ▶ RAM
 - ▶ batterie
 - ▶ stockage
 - ▶ connectivité

- ▶ architecture différente de x86
 - ▶ micro-contrôleurs
 - ▶ JavaCard
 - ▶ ARM

Qu'est-ce que l'embarqué

Caractérisation de l'embarqué

- ▶ ressources limitées
 - ▶ CPU
 - ▶ RAM
 - ▶ batterie
 - ▶ stockage
 - ▶ connectivité

- ▶ architecture différente de x86
 - ▶ micro-contrôleurs
 - ▶ JavaCard
 - ▶ ARM

- ▶ environnement de développement spécifique
 - ▶ souvent très intégrés
 - ▶ parfois propriétaires

Exemples de plateformes (1/4)

Cartes à puces JavaCard

- ▶ restrictions assez fortes sur de tous les aspects
- ▶ langage de programmation proche de Java
- ▶ outils de déploiement souvent propriétaires
- ▶ format « binaire » standard (le *bytecode*)

Exemples de plateformes (1/4)

Cartes à puces JavaCard

- ▶ restrictions assez fortes sur de tous les aspects
- ▶ langage de programmation proche de Java
- ▶ outils de déploiement souvent propriétaires
- ▶ format « binaire » standard (le *bytecode*)

- ▶ plusieurs mécanismes de sécurité

Exemples de plateformes (1/4)

Cartes à puces JavaCard

- ▶ restrictions assez fortes sur de tous les aspects
- ▶ langage de programmation proche de Java
- ▶ outils de déploiement souvent propriétaires
- ▶ format « binaire » standard (le *bytecode*)

- ▶ plusieurs mécanismes de sécurité

- ▶ très employé pour les composants de confiance (« *secure elements* »)

Exemples de plateformes (2/4)

Arduino

- ▶ restrictions fortes sur la plupart des aspects
- ▶ architecture spécifique (micro-contrôleurs Atmel pouvant reposer sur des cœurs ARM)
- ▶ environnement de développement libre en C
- ▶ nombreux accessoires disponibles

Exemples de plateformes (2/4)

Arduino

- ▶ restrictions fortes sur la plupart des aspects
- ▶ architecture spécifique (micro-contrôleurs Atmel pouvant reposer sur des cœurs ARM)
- ▶ environnement de développement libre en C
- ▶ nombreux accessoires disponibles

- ▶ certains modèles disposent d'une MPU ou d'une MMU

Exemples de plateformes (2/4)

Arduino

- ▶ restrictions fortes sur la plupart des aspects
- ▶ architecture spécifique (micro-contrôleurs Atmel pouvant reposer sur des cœurs ARM)
- ▶ environnement de développement libre en C
- ▶ nombreux accessoires disponibles

- ▶ certains modèles disposent d'une MPU ou d'une MMU

- ▶ idéal pour des TP pour de l'embarqué

Exemples de plateformes (3/4)

Raspberry Pi

- ▶ petits ordinateurs avec une connectique riche
- ▶ architecture ARM complète
- ▶ systèmes Linux complets (Debian, Arch Linux, Android...) ou Windows 10 IoT pour les modèles récents

Exemples de plateformes (3/4)

Raspberry Pi

- ▶ petits ordinateurs avec une connectique riche
- ▶ architecture ARM complète
- ▶ systèmes Linux complets (Debian, Arch Linux, Android...) ou Windows 10 IoT pour les modèles récents
- ▶ richesse des mécanismes de sécurité (MMU, TrustedZone, XN...)

Exemples de plateformes (3/4)

Raspberry Pi

- ▶ petits ordinateurs avec une connectique riche
- ▶ architecture ARM complète
- ▶ systèmes Linux complets (Debian, Arch Linux, Android...) ou Windows 10 IoT pour les modèles récents
- ▶ richesse des mécanismes de sécurité (MMU, TrustedZone, XN...)
- ▶ intéressant pour des TP
- ▶ mais très (trop?) proche d'un ordinateur classique

Exemples de plateformes (4/4)

Téléphones et tablettes Android

- ▶ essentiellement des ordinateurs classiques
- ▶ architecture ARM
- ▶ batterie limitée
- ▶ beaucoup d'interfaces réseau
- ▶ système Android (distribution Linux complète)

Exemples de plateformes (4/4)

Téléphones et tablettes Android

- ▶ essentiellement des ordinateurs classiques
- ▶ architecture ARM
- ▶ batterie limitée
- ▶ beaucoup d'interfaces réseau
- ▶ système Android (distribution Linux complète)

- ▶ nombreux mécanismes de sécurité (matériels et logiciels)

Exemples de plateformes (4/4)

Téléphones et tablettes Android

- ▶ essentiellement des ordinateurs classiques
- ▶ architecture ARM
- ▶ batterie limitée
- ▶ beaucoup d'interfaces réseau
- ▶ système Android (distribution Linux complète)

- ▶ nombreux mécanismes de sécurité (matériels et logiciels)

- ▶ mêmes remarques que pour Raspberry Pi
- ▶ coût élevé

Avertissement sur les recommandations

La suite de cet exposé comporte quelques recommandations

Mon expérience concrète sur le sujet est néanmoins très réduite.
Restez donc critique !

Plan

Définitions de l'embarqué

JavaCard

Arduino

Android

Quelques recommandations générales

Déploiement du code

Le déploiement d'une application JavaCard suit généralement le cheminement suivant

- ▶ développement en Java (sous-ensemble compatible)
- ▶ compilation vers du *bytecode* JavaCard
- ▶ vérification du *bytecode*
- ▶ signature du *bytecode*
- ▶ injection sur la carte avec vérification de la signature

Déploiement du code

Le déploiement d'une application JavaCard suit généralement le cheminement suivant

- ▶ développement en Java (sous-ensemble compatible)
- ▶ compilation vers du *bytecode* JavaCard
- ▶ vérification du *bytecode*
- ▶ signature du *bytecode*
- ▶ injection sur la carte avec vérification de la signature

Mécanismes de sécurité

- ▶ le *firewall* entre applications (parfois perméable)
- ▶ la vérification de *bytecode* (faite hors carte)
- ▶ le verrouillage du chargement d'applications (pas toujours fait)

Exécution en milieu hostile

La dure vie d'un *secure element*

- ▶ l'attaquant contrôle l'alimentation, l'horloge
- ▶ l'attaquant peut surveiller la consommation et le rayonnement
- ▶ l'attaquant peut s'attaquer physiquement à la carte

Exécution en milieu hostile

La dure vie d'un *secure element*

- ▶ l'attaquant contrôle l'alimentation, l'horloge
- ▶ l'attaquant peut surveiller la consommation et le rayonnement
- ▶ l'attaquant peut s'attaquer physiquement à la carte

Attaques envisageables

- ▶ attaques par canaux auxiliaires (SPA, DPA, etc.)
- ▶ attaques par injection de faute

Exécution en milieu hostile

La dure vie d'un *secure element*

- ▶ l'attaquant contrôle l'alimentation, l'horloge
- ▶ l'attaquant peut surveiller la consommation et le rayonnement
- ▶ l'attaquant peut s'attaquer physiquement à la carte

Attaques envisageables

- ▶ attaques par canaux auxiliaires (SPA, DPA, etc.)
- ▶ attaques par injection de faute

Ressources sur ce point

- ▶ fiches sur les composants
 - ▶ <http://cyberedu.fr/billets/2016/07/1a-mallette-cyberedu/>
- ▶ projet SeseLab présenté à RESSI 2018
 - ▶ <https://ressi2018.sciencesconf.org/187696>

Aspects propriétaires et limites du modèle

L'accès complet aux cartes est parfois complexes

- ▶ besoin d'accepter des NDA (accords de non-divulgence) pour accéder à toutes les fonctionnalités
- ▶ aucun accès aux fonctionnalités natives
- ▶ cependant, ces éléments sont importants pour la sécurité

Plan

Définitions de l'embarqué

JavaCard

Arduino

Android

Quelques recommandations générales

Contraintes sur la plateforme

Quelques contraintes sur la plateforme

- ▶ ressources assez limitées (certains algorithmes ne sont pas envisageables)
- ▶ prise en main de l'environnement de développement et de déploiement
- ▶ absence de système d'exploitation

Contraintes sur la plateforme

Quelques contraintes sur la plateforme

- ▶ ressources assez limitées (certains algorithmes ne sont pas envisageables)
- ▶ prise en main de l'environnement de développement et de déploiement
- ▶ absence de système d'exploitation

Avantage pour un premier pas dans l'embarqué

- ▶ plateforme abordable et facile d'accès
- ▶ nombreux livres et projets

Quid de la sécurité ?

Pas ou peu de mécanismes de sécurité

- ▶ pas toujours de MMU ni de MPU, donc pas de protection matérielle
- ▶ code tournant souvent à même le CPU (*bare metal*), donc pas de protection logicielle

Bonnes pratiques de développement

Pour mettre au point le code, et assurer des propriétés de sécurité

- ▶ activer les avertissements du compilateur C
- ▶ durcir la chaîne de compilation lorsque c'est possible
- ▶ toujours vérifier les entrées
 - ▶ y compris celles qui ne semblent pas contrôlées par l'utilisateur, comme les capteurs!
- ▶ comprendre la réalité des projections mémoire
 - ▶ zones en lecture seule
 - ▶ zones I/O « volatiles »
 - ▶ zones de mémoire persistantes

Plan

Définitions de l'embarqué

JavaCard

Arduino

Android

Quelques recommandations générales

Bonnes pratiques de développement

Spécificités d'Android

- ▶ code natif
 - ▶ pas portable
 - ▶ qui échappe essentiellement à l'analyse de code

Bonnes pratiques de développement

Spécificités d'Android

- ▶ code natif
 - ▶ pas portable
 - ▶ qui échappe essentiellement à l'analyse de code
- ▶ modèle ouvert / périmètre flou
 - ▶ pas de point d'entrée clairement identifié
 - ▶ couplage fort entre une appli web et son service web associé

Impact de l'architecture sur la sécurité

Dépendances à maîtriser dans le temps

- ▶ pérennité du site web
- ▶ gestion des certificats X.509
- ▶ pérennité du nom de domaine

Plan

Définitions de l'embarqué

JavaCard

Arduino

Android

Quelques recommandations générales

Exposition

En sécurité, il faut toujours se poser la question de la surface d'attaque

- ▶ étudier les interfaces de communication
- ▶ prendre en compte les capteurs
- ▶ dans certains milieux hostiles, durcir le code pour prendre en compte la menace physique

Infrastructure

Dans l'embarqué, l'équipement considéré n'est généralement pas seul

- ▶ dépendance à des capteurs externes
- ▶ relation à d'autres objets ou consoles d'administration
- ▶ lien fort avec des serveurs distants, notamment pour les calculs

Infrastructure

Dans l'embarqué, l'équipement considéré n'est généralement pas seul

- ▶ dépendance à des capteurs externes
- ▶ relation à d'autres objets ou consoles d'administration
- ▶ lien fort avec des serveurs distants, notamment pour les calculs

Tous ces éléments doivent être pris en compte

- ▶ analyse de risque
- ▶ protection des communication
- ▶ vérification d'intégrité ou de cohérence lorsque c'est possible

Durcissement

Lorsque c'est possible, appliquer des mécanismes de défense en profondeur

- ▶ protections mémoire
- ▶ réduction des privilèges
- ▶ programmation défensive
- ▶ de manière générale, bonnes pratiques de développement

Mise à jour ?

Utile dès qu'un équipement

- ▶ est connecté à quelque chose non maîtrisé
- ▶ réalise un traitement non trivial sur des données

Mise à jour ?

Utile dès qu'un équipement

- ▶ est connecté à quelque chose non maîtrisé
- ▶ réalise un traitement non trivial sur des données

À prendre en compte dès le départ

- ▶ impact non négligeable sur l'architecture logicielle
- ▶ généralement difficile voire impossible après coup
- ▶ la fonctionnalité de mise à jour ne doit pas affaiblir la sécurité !

Ultime avertissement

Je ne suis pas spécialiste de l'embarqué

- ▶ ces conseils sont des premiers éléments
- ▶ ils sont à prendre avec un esprit critique

Questions ?

Merci pour votre attention