Livre blanc

Partie 2 :

Sandbox pour une meilleure sécurité de l'IoT

Utilisez Legato® pour renforcer la sécurité, réduire les risques et combattre les attaques

La plateforme open source Legato a été intégrée sur nos modules de la série smart et est entièrement testée et validée pour les séries AirPrime® WP et AR.

Les développeurs de dispositifs IoT peuvent utiliser des applications « sandboxed » au sein de la plateforme open-source intégrée Legato, afin de minimiser l'impact d'un certain nombre de menaces de sécurité, allant des logiciels malveillants et du piratage aux erreurs de code involontaires.

Dans l'Internet des Objets (IoT) d'aujourd'hui, les appareils effectuent souvent un certain nombre de tâches complexes, chaque tâche nécessitant plusieurs processus et divers morceaux de code exécutable. Un dispositif IoT pour le suivi des véhicules, par exemple, peut accomplir plusieurs tâches, notamment surveiller la pression des pneus, suivre la vitesse et documenter les comportements de conduite erratiques. Il peut également être équipé pour prendre en charge des éléments tels que le GPS pour la localisation, l'appel d'urgence électronique pour l'assistance ou un point d'accès WI-FI dans la voiture. Chacune de ces fonctions (pression, vitesse, comportement, GPS, eCall, hotspot) est essentiellement une application distincte, avec son propre ensemble de tâches à effectuer et son propre logiciel à assembler, déboguer et gérer. Rassembler toutes ces applications séparées, afin qu'elles fonctionnent efficacement dans un seul appareil, peut être un défi complexe étant donné que les développeurs de l'IoT travaillent généralement dans des délais serrés et des contraintes budgétaires strictes. La pression pour fournir des produits riches en fonctionnalités aussi rapidement et à moindre coût que le marché le demande signifie que les développeurs ont besoin d'outils qui aident à assurer un fonctionnement sûr et robuste.

L'un de ces outils est le cadre d'application Legato à l'intérieur de la plate-forme embarquée Legato, un projet Linux à source ouverte créé spécifiquement pour l'IoT. Legato utilise une technique appelée « sandboxing », pour isoler les applications et les confiner dans un environnement spécifique et sécurisé. Le « sandboxing » permet de configurer des fonctionnalités avancées tout en aidant à contenir les erreurs, à renforcer les vulnérabilités et à réduire les risques.

Ce document technique présente les concepts de base du sandboxing Legato et explique comment Legato peut minimiser l'impact si un appareil est victime d'une attaque. Nous commençons par un examen des vulnérabilités du système que Legato est conçu pour aider les développeurs à éviter.

Partie 2 :
Sandbox pour une meilleure sécurité de l'IoT

1. Source de vulnérabilité de l'IoT

D'une manière générale, les dispositifs IoT collectent des données et déclenchent des actions. Si quelque chose ne va pas dans ce processus, un certain nombre de problèmes peuvent survenir. Le dispositif peut se reconfigurer, fournir un accès non autorisé à des informations privées, consommer toutes ses ressources disponibles ou faire courir un risque d'attaque à l'ensemble du déploiement. Ces problèmes peuvent souvent être attribués à l'une des trois sources suivantes : un logiciel corrompu qui contient une sorte logiciel malveillant, un logiciel qui utilise des interfaces externes qui n'ont pas été correctement sécurisées, ou un logiciel qui présente des bogues invisibles ou des erreurs de logique.

- Applications tierces contaminées (malware)

La plupart des développeurs de logiciels ont de bonnes intentions et ne créent pas sciemment des applications qui endommageront le système ou l'empêcheront de fonctionner correctement, mais les codeurs malveillants existent. Il suffit de peu de choses pour injecter un logiciel malveillant dans un bon logiciel. Une bombe à fourche, par exemple, qui est un type d'attaque par déni de service, peut être écrite avec seulement quelques lignes de code, et ces lignes peuvent être relativement faciles à cacher dans un long morceau de code d'application. Les auteurs de logiciels malveillants veillent souvent à ce que leurs logiciels paraissent authentiques et fiables, de sorte qu'il peut être difficile de savoir ce qui est acceptable. Par conséquent, l'installation d'une application tierce peut également signifier l'intégration de logiciels malveillant dans votre système, même si cette application a été inspectée et testée visuellement.

- Des interfaces externes mal sécurisées

Chaque fois qu'un dispositif IoT communique avec le monde extérieur (pour envoyer des données, recevoir des instructions ou interagir avec un système distant) il y a un risque d'attaque. En effet, le réseau utilisé pour la communication peut, dans certaines circonstances, être accessible à un pirate informatique ou à un logiciel malveillant. Le pirate n'a pas besoin d'avoir un accès physique à l'appareil, puisque l'attaque a lieu quelque part sur le chemin de communication. Si le dispositif IoT n'a pas été équipé des protections nécessaires sur les interfaces externes, un attaquant pourrait potentiellement avoir accès au dispositif IoT et, par extension, à d'autres dispositifs du déploiement.

- Les erreurs de code commises par inadvertance

Les fautes de frappe, les déconnexions dans la logique et autres gaffes sont une réalité, et les erreurs passent parfois inaperçues pendant la phase de débogage. Quelque chose de simple, comme une condition de course involontaire ou boucle infinie, peut accrocher le système et vider la batterie, ou causer d'autres problèmes inattendus. De plus, les développeurs n'ont pas toujours le temps ou l'expertise nécessaire pour identifier tous les dangers imaginables, ce qui signifie qu'un problème peut passer inaperçu jusqu'à ce qu'il soit déclenché par une combinaison particulière d'événements.

Legato aide les développeurs à éviter ces problèmes et à minimiser leur impact potentiel. Conçu dès le départ pour être utilisé dans l'IoT, Legato rend l'appareil plus robuste et moins vulnérable aux logiciels malveillants, au piratage et aux erreurs de code involontaires, grâce à un certain nombre de techniques. L'une de ces techniques est une méthode appelée « sandboxing », qui place chaque application dans son propre environnement isolé.

Le reste de ce document examine l'approche de Legato en matière de sandboxing

2. Sandboxing et sécurité

Historiquement, les systèmes embarqués se sont concentrés sur la fonctionnalité, la robustesse et l'efficacité, mais n'ont pas ou peu pensé à la sécurité. Ce manque d'attention à la sécurité s'est poursuivi alors même que les systèmes embarqués sont devenus de plus en plus complexes et que les risques augmentent de manière exponentielle à mesure que nous entrons dans l'ère de l'IoT où a connectivité est omniprésente. En outre, le passage d'applications fonctionnant sur des RTOS en métal nu et de petite taille à des systèmes d'exploitation complets à usage général entraîne une augmentation considérable des risques de sécurité.

Le « sandboxing » est une technique utilisée pour atténuer les risques de sécurité en isolant les tâches les - unes des autres, ce qui aide de la manière suivante :

1- Une mauvaise exécution des tâches ne peut pas nuire à d'autres tâches. Par exemple, si une tâche chargée de faire jouer de la musique dans une voiture est compromise, elle ne devrait jamais pouvoir passer outre les freins de la voiture.

2- Une mauvaise gestion des tâches ne peut pas endommager le système. Par exemple, une tâche isolée avec des contrôles appropriés ne devrait pas pouvoir consommer autant de mémoire que le système ne réagit plus.

3- La complexité est contrôlée par la modularité. Tout comme dans la programmation où des morceaux de code apparentés sont regroupés pour former un module, une classe, etc., les différentes tâches doivent être isolées les unes des autres et la communication inter-tâches limitée à des interfaces bien définies. La réduction de la complexité est bénéfique pour la sécurité :

- Faciliter le développement d'un code de sécurisé. De nombreuses vulnérabilités de codage sont introduites parce qu'il est très difficile de prévoir et de vérifier toutes les configurations et tous les chemins de code possibles dans un système complexe.
- Une complexité moindre rend les audits de sécurité beaucoup plus faciles à réaliser.

La plupart des techniques de sandboxing utilisent la virtualisation, qui consiste à faire fonctionner plusieurs systèmes d'exploitation sur la même plateforme grâce à l'utilisation d'un hyperviseur. Les tâches qui doivent être isolées les unes des autres s'exécutent dans des instances de systèmes d'exploitation distinctes. Cette méthode est très gourmande en processeur et en mémoire et ne convient généralement pas à la plupart des dispositifs de l'IoT.

Les sandboxing Legato, en revanche, utilisent un seul système d'exploitation, ce qui les rend beaucoup plus légers. Les tâches sont accomplies par des applications qui peuvent exécuter plusieurs processus. Chaque application fonctionne dans son propre sandboxing et la communication inter-applications n'est possible qu'à travers des interfaces bien définies qui sont authentifiées dans le cadre de Legato.

Le cadre Legato est un ensemble d'outils, de bibliothèques et de processus d'exécution conçus spécialement pour faciliter et sécuriser le développement, le déploiement et la gestion des applications IoT.

3. Comment sandboxing Legato fonctionne

L'environnement d'exécution du cadre d'application Legato est conçu pour un dispositif Linux embarqué et se compose d'un certain nombre de processus de démon qui gèrent le cycle de vie des applications. Le démon du framework qui est le plus directement responsable de la mise en place des sandboxes est un démon privilégié appelé le superviseur. Lorsqu'il lance une application, le superviseur doit d'abord créer la sandbox et ensuite lancer tous les processus de l'application à l'intérieur de la sandboxe.

Nous allons maintenant examiner ces deux étapes en détail. Notez que des erreurs subtiles dans les procédures ci-dessous peuvent entraîner des atteintes à la sécurité désastreuses. Si cette section vous semble trop complexe, passez à la section IV car le cadre Legato fait tout cela automatiquement et fournit des outils pour rendre la construction d'applications sandboxes et direct

3.1. Créer la sandbox

Le sandbox lui-même est un emplacement dans le système de fichiers qui contient tous les fichiers dont l'application a besoin, mais pas plus. Pour créer cette zone de sandbox, l'utilisateur devra suivre les étapes suivantes :

a. Créer un système de fichiers isolé pour agir comme la racine du sandbox. Par exemple, nous pourrions utiliser :

"/tmp/legato/sandboxes/helloworld" pour une application helloWorld.

```
mkdir("/tmp/legato/sandboxes/helloWorld", S_IRWXO);
```

b. Pour garantir la non-persistance du sandbox, un système de fichiers basé sur la RAM est montré à la racine du sandbox .tmpfs est utilisé plutôt que ramfs car tmpfs peut être configuré pour croître jusqu'à une limite de taille spécifique. Ceci est important pour contrôler la taille maximale à laquelle l'application peut faire croître son système de fichiers.

```
char opt[OPTION_STR_SIZE];

snprintf(opt, sizeof(opt), "size=%d,mode=%.4o,uid=0,gid=0",
                fileSysSizeLimit, S_IRWXO);

mount("helloWorldSandbox", sandboxPath, "tmpfs", MS_NOSUID, opt);
```

c. Importez les fichiers dont l'application aura besoin dans le sandbox vide. Les fichiers d'application tel que les exécutables, les bibliothèques et les fichiers de ressources qui sont disponibles sur le système au moment de l'installation de l'application doivent encore être mis à disposition dans la sandbox. Seuls les fichiers que nous importons dans le sandbox seront visibles/accessibles par l'application. Tous les autres fichiers du système seront complétement invisibles pour l'application en cours d'exécution. Cela nous permet de satisfaire au principe de sécurité du « moindre « privilège ».

Bien qu'il soit possible de copier simplement les fichiers dans le sandbox, il faudrait une deuxième copie de fichier, ce qui est assez gaspilleur. Une solution plus optimale consiste à créer des liens à l'intérieur de la sandbox vers les fichiers externes. Cependant, les liens en dur ne sont pas possibles car ils ne peuvent pas traverser les systèmes de fichiers et les liens symboliques ne fonctionneront pas car ils ne se résoudront pas à travers un « chroot » (plus d'informations sur le chroot dans le section suivante). Legato utilise donc un montage de liaison pour faire disparaître le même fichier à différents endroits. Par exemple, pour importer /dev/null dans le sandbox :

```
/* Create a mount point for the file inside the sandbox.*/

char destPath[] = "/tmp/legato/sandboxes/helloWorld/dev/null";

creat(destPath, S_IRUSR);

 /* Bind mount the original file at the location inside the sandbox, to make the file accessible from both locations. */

mount("/dev/null", destPath, NULL, MS_BIND, NULL);
```

Cela peut être répété pour tous les dossiers dont la demande aura besoin. Il faut veiller à ce que les dossiers aient les autorisations appropriées. Par exemple, les fichiers exécutables ne doivent pas être accessibles en écriture par l'application. Les fichiers exécutables ne doivent pas non plus avoir leurs bits de permission setuid réglés, car cela peut entraîner de dangereuses escalades de privilèges.

A ce stade, notre sandbox est complet et prêt pour les procédures de candidature.

3.2. L'exécution des processus de candidature dans le sandbox

Les applications peuvent avoir un ou plusieurs programmes exécutables et il peut y avoir plusieurs processus en cours d'exécution simultanément dans une application. Pour lancer chaque processus d'application, le superviseur Legato utilise la procédure suivante pour non seulement les processus mais aussi pour confiner les processus dans le sandbox précédemment créée.

a- Créer un nouvel utilisateur et un nouveau groupe pour l'application. Il est important que l'application ne s'exécute pas en tant que root car cela donnera à l'application des privilèges de super-utilisateur et permettra à l'application de s'échapper de la sandbox.

```
uid_t uid;

gid_t gid;

user_Create("helloWorld", &uid, &gid);
```

b- Le contrôleur bifurquera automatiquement une instance de l'enfant qui deviendra plus tard la procédure de demande.

```
pid_t pid = fork();

if (pid == 0)

 {

        /* This is the child process code. */

}

else

 {

/* This is the parent process (Supervisor) code. */

 }
```

c- Effacer les variables d'environnement héritées du parent en utilisant l'appel système clearenv(). Définir les variables d'environnement appropriés pour le processus de demande en utilisant l'appel seten(). Notez que les variables d'environnement du processus de demande sont définies à l'aide du fichier manifeste d'application Legato (.adef).

d- Remplacez le répertoire de travail du processus (chdir) et le répertoire racine de son système de fichiers (chroot) par la racine du répertoire de la sandbox.

/ Change working directory. */*

chdir("/tmp/legato/sandboxes/helloWorld");

/ Chroot to the sandbox. */*

chroot("/tmp/legato/sandboxes/helloWorld")

Chroot modifie simplement ce que le processus considère comme son répertoire racine. Par exemple, à ce stade, si le processus devait créer un fichier « /greeting.txt » et lister tous les fichiers à « / », il verrait « greeting.txt ». Mais pour un processus sans sandbox, le même fichier apparaîtrait à « /tmp/legato/sandboxes/helloWorld/greeting.txt ». En fait, Legato a piégé le processus de sorte qu'il ne peut voir que les fichiers qui se trouvent sous sans sandbox.

Il est également important de changer le répertoire de travail avec chdir pour qu'il soit dans le sandbox. Sans cette étape, le processus aurait un chemin résolu vers le monde extérieur. Par exemple, si le processus confiné appelait :

/ Try to move back up to the real root (outside the sandbox). */ chdir("../../../../../../../../../..");*

Il resterait encore dans le sandbox. Ce ne serait pas le cas si nous n'effectuions pas l'appel chdir() initial.

e- Remplacez les identifiants de groupe et d'utilisateur du processus par ceux créés à l'étape i. Cela dépouillera le processus de toutes ses capacités et l'empêchera de s'échapper de son sandbox

/* Effacer la liste de nos groupes supplémentaires. */ setgroups(0, NULL) ;

/* Définissez l'ID du groupe primaire pour le processus. */ setgid(gid) ;

/* Définit l'ID de l'utilisateur pour le processus. Cela permet de définir tous les ID utilisateur (réel, effectif, enregistré) et d'effacer toutes les capacités. Cette fonction DOIT être appelée après tous les appels système précédents, car une fois qu'elle est effectuée, les privilèges de root sont perdus. */ setuid(uid) ;

f- A ce stade, le processus d'application est confiné en toute sécurité dans la sandbox et peut exécuter le code de l'application en appelant simplement l'appel système exec().

/ Exécute le fichier exécutable du processus avec les paramètres de ligne de commande appropriés. */*

execvp(fichier exécutable, cmdLineParams) ;

Le processus confiné fait maintenant fonctionner le code d'application dans son sandbox.

Remarque : l'ordre des étapes ci-dessus est extrêment important. De même, les extraits de code ci-dessus ne sont qu'indicatifs et peuvent ne pas réfleter l'état actuel du projet Legato. Par souci de concision, le code de vérification des erreurs n'a pas été inclus dans les extraits de code ci-dessus. Cela ne signifie pas que la vérification des erreurs n'est pas importante, au contraire, la vérification des codes de retour et le traitement des conditions d'erreur est l'un des aspects les plus importants des pratiques de codage sûres.

Bien qu'ils ne soient pas couverts par ce livre blanc, les sandboxes Legato utilisent également des cgroupes et des limites pour faire respecter les limites de ressources et SMCK comme système de contrôle d'accès obligatoire. Le contrôle d'accès obligatoire est utilisé pour renforcer les sandboxes.

4. Création et application sandbox Legato

Les applications Legato sandbox sont mises en boîte par défaut et ne nécessitent aucun effort supplémentaire de la part du développeur de l'application. Il suffit de créer un fichier manifeste, appelé .adef, pour l'application.

Examinons un exemple d'application incluse dans Legato repo, appelée hellPipe. Cette application consiste en un script bash appelé « utilPipes », qui exécute ls dans le répertoire du sandbox et l'envoie au grep :

```
#!/bin/sh

# Script simple qui montre la tuyauterie entre les utilitaires. echo "Starting script"

ls / | grep "usr"

echo "Done script"
```

Vous trouverez ci-dessous le fichier ShellPipe.adeg qui est le manifeste de cette application et qui se compose de trois sections : liasses, exigences et processus.

```
des liasses :

{
        fichier :
        {
                // Regroupement du script dans le sandbox avec les autorisations d'exécution.   [rx]
                utilPipes /
        }
}

exigences :

{
        fichier :
        {
                // Ajout des utilitaires sh (shell), ls, grep et echo dans la sandbox.
                 /bin/sh /usr/local/bin/
                 /bin/ls /usr/local/bin/
                /bin/grep /usr/local/bin/
                /bin/echo /usr/local/bin/
        }
}
```

```
Processes :
{
        run :
        {
                // Exécutez sh (shell) et passez-lui le script pour qu'il s'exécute.   (sh utilPipes)
        }
}
```

Définir les fichiers qui doivent être inclus dans le paquet d'installation de l'application et qui doivent être mis à la disposition de l'application. Dans l'exemple ci-dessus, le script « utilPipes » est fourni avec l'application et indique au cadre de travail d'importer le script dans la sandbox à la racine de la sandbox.

Obligatoire : définir les fichiers nécessaires à l'application mais déjà présents dans le système et qui n'oont pas besoin d'être empaquets avec l'application. Dans l'exemple ci-dessus, les utilitaires « sh », « Is », « grep » et « echo » doivent être importés dans le sandbox de l'application sous « /url/local/bin/ » (Notez que « /usr/local/bin/ » est relatif à la racine du sandbox.

Désormais, lorsque l'application sera lancée, elle sera automatiquement mise en sandbox. Notez que le script s'exécutera dans le sandbox, en listant les fichiers et les répertoires qui se trouvent à la racine du sandbox. Si le script utilPipes devait être exécuté en dehors du sandbox, il énumérerait les fichiers et les répertoires de la racine du système.

Conclusion

Sandboxing est un élément très important de la création d'applications IoT. Il permet de séparer et de protéger les processus de demande, ce qui facilite leur maintenance et leur contrôle. Les commandes Legato pour travailler avec les sandbox sont simples et faciles à utiliser, et conçues pour donner aux développeurs la flexibilité dont ils ont besoin pour garantir un système robuste. Le résultat final est un déploiement de l'IoT qui est plus résistant et plus susceptible de résister aux effets néfastes des logiciels malveillants, du piratage et des bogues.