Les architectures de sécurité des OS mobiles

Cette suite d'articles présente les différents achitectures de sécurité proposé par Android, IOS et Windows Phone. Puis décrit les différents attaques et comment s'en prémunir.

Par Philippe PRADOS - 2012
www.prados.fr

La sécurité des applications mobiles est un challenge pour les développeurs. Pour appréhender correctement ce sujet, il faut être conscient de la façon dont votre code utilise l'information et vérifier qu'il assure une sécurité adaptée. Par exemple :

- Des données personnelles appartenant à vos utilisateurs sont confié à votre logicielle. Il est de votre responsabilité de garder ces données à l'abri des regards indiscrets. En outre, il est de votre responsabilité de vous assurer que votre logiciel ne recueille que les renseignements qu'il exige.
- Si votre logiciel accède à Internet ou à des fichiers qui pourraient avoir été préalablement envoyé à quelqu'un sur Internet, il est de votre responsabilité de vous assurer de consulter ces fichiers de façon sûre, que ces derniers ne fournissent pas un vecteur pour accéder à d'autres données personnelles qui peuvent être stockés sur l'appareil mobile.
- Si votre logiciel transmet des informations personnelles sur Internet, il est de votre responsabilité de le faire de façon sécuritaire pour prévenir l'accès non autorisé ou la modification des données en transit.
- Si votre logiciel permet d'accéder aux données signées, il est de votre responsabilité de vérifier ces signatures afin de s'assurer que les données n'ont pas été altérées.

Dans la phase de planification, vous devez déterminer la nature des menaces à votre logiciel et architecturer votre code de telle manière à maximiser la sécurité. Pour ce faire, vous devez construire un modèle de menace qui expose les manières dont votre logiciel peut être attaqué.

Pour évaluer le risque que votre code poserait s'il était compromis, vous devrez d'abord supposer que votre programme va être attaqué. La quantité de temps et d'effort que l'attaquant passera à attaquer votre programme dépend de facteurs tels que :

- La valeur des données de votre programme gère. Mémorise t'il de numéros de carte de crédit ?
- La fiabilité et la sécurité des entreprises qui fournissent des services et des données que votre code utilise.
- Les profils des clients utilisant votre programme. Particulier ? Militaire ?
- La diffusion du programme. Est-il un utilisé par un seul petit groupe de travail ou par le monde entier ?

Basé sur ces mêmes facteurs, vous devez décider quel niveau de risque est acceptable. Une perte de données, qui coûtera 1000 € de votre entreprise pour y remédier ne pas justifie pas un effort de développement de 10 000 € pour fermer toutes les failles de sécurité potentiels. D'autre part, des dommages à la réputation de votre entreprise pourrait valoir beaucoup plus à le long terme que cela ne coûterait de concevoir et développer un code sécurisé.

Les attaques s'occupe généralement des mêmes cibles :

- Les entrées du programme. Si un attaquant peut provoquer un débordement de tampon, ils pourraient être en mesure d'exécuter son propre code ou compromettre les

données de l'utilisateur ou du système ;
- Les sorties du programme (à l'utilisateur ou à un autre module du logiciel). L'attaquant pourrait être en mesure d'accéder à des informations privées stockées sur le système, ou de lire et modifier les informations étant transmises entre les modules ;
- Les données stockées sur le système (de façon permanente, comme dans une base de données, ou temporaire, comme dans une variable globale). Ces données pourraient être volées et envoyées à un attaquant, modifiées par un pirate ou compromises ;
- L'environnement du programme. L'environnement d'exécution d'un programme comprend ses descripteurs de fichiers ouverts, les variables d'environnement, les ports de réseaux, les fichiers de préférences, etc.

La sécurité en mobilité est assez différentes des approches traditionnelles pour plusieurs raisons :

- Les applications doivent être isolées les unes des autres ;
- Elles doivent se protéger les uns des autres ;
- Les informations sensibles ne doivent pas être compromises en cas de vols du terminal ;
- Les performances et les ressources sont limitées ;
- Le réseau est chaotique avec une forte latence ;
- Les tests 3G doivent s'effectuer directement sur Internet, avant d'avoir complètement protégé et qualifié le service.

Les modèles de sécurité

Plusieurs systèmes d'exploitations sont proposées pour la mobilité. Certains sont dérivés de système existant, d'autres ont été conçu spécifiquement. Chaque système a fait des choix différents pour répondre aux besoins de sécurités.

Débloquer son téléphone

Il existe de nombreuses techniques et outils pour débloquer un téléphone. Cela casse la garantie constructeur mais permet, pour nous les geeks, de reprendre la main sur le terminal. Il est ainsi possible d'installer toute les applications que l'on souhaite, de modifier le petit détail qui gène, de renforcer la sécurité parfois. Mais, il faut bien avoir conscience qu'un téléphone déverrouillé est un téléphone fortement affaiblit. Si vous pouvez tous contrôler, des applications ou un voleur peuvent faire de même. Comme dirais Jean-Pierre Troll : « C'est la porte ouverte à toute les fenêtres » ! Donc prudence, prudence...

Modèle de sécurité Android

Nous allons présenter le modèle de sécurité choisi par Android.

Isolation

Comme tous les systèmes mobiles, Android propose un modèle de sécurité basé sur l'isolation des applications. Il utilise pour cela des utilisateurs Linux différents par application. De plus, il applique une modification spécifique au noyau Linux pour autoriser certaines APIs sensibles (comme l'accès au réseau) à des utilisateurs spécifiques systèmes, codé en dur et inaccessibles aux applications.

La mémoire utilisé par un processus n'est pas accessible aux autres processus.

Chaque application possède un répertoire privé, dont l'accès est limité à l'utilisateur associé à l'application. Il est possible de partager le même utilisateur entre plusieurs applications, a condition de partager également la même signature. Il est également possible d'exposer certains fichiers aux autres applications, en modification les droits d'accès de l'OS ou en

exposant des APIs spécifiques.

Les applications Android n'ont pas accès aux périphériques (/dev/*), ce qui interdit une utilisation directe du hardware. Elles doivent passer par un processus spécifique (system_app) via une communication inter-processus. Ce mécanisme utilise un driver noyau spécifique qui injecte l'identification de l'application ainsi que le groupe de l'appelant (uid et gid) . system_app peut alors vérifier les privilèges de l'appelant avant d'effectuer les traitements sensibles.

Le système est répartie dans plusieurs partitions. La partition système est monté en lecture seule. Il est possible de démarrer le système en mode « sauvegarde ». Dans ce mode, seule les applications initiales sont activées.

Chaque application peut exprimer les privilèges dont elle a besoin. Certains privilèges ne peuvent être accordés qu'aux applications systèmes ou aux applications signées par la signature de la plate-forme (le fournisseur du téléphone). Ainsi, les constructeurs peuvent proposer des applications systèmes sensibles (comme l'exposition de l'écran du terminal via le réseau, l'installation silencieuses d'applications), simplement en publiant une application sur la place de marché.

Une application peut proposer de nouveaux privilèges. Ce qui permet à cette dernière ou à d'autres applications de les utiliser. Les nouveaux privilèges peuvent être accordés à toutes les applications ou limités aux applications ayant la même signature numérique. Cela permet la communication entre des applications ou des extensions d'une application par le même auteur, sans compromettre la sécurité.

Par exemple, l'entreprise Acme propose une application A permettant d'enregistrer un compte et propose le nouveau privilège « acces au mot de passe de Acme ». Une application X ne peut obtenir ce privilège pour récupérer le secret, si elle n'est pas signé par Acme. Par contre, Acme peut proposer une application B, signé avec le même certificat. Cette dernière peut ainsi obtenir le privilège d'obtenir le mot de passe. Ainsi, il n'est pas nécessaire de le demander pour chaque application de Acme.

Les applications peuvent être rédigées en Java ou en C/C++. Cela n'a aucun impact sur la sécurité. En effet, chaque application Java peut invoquer un code natif et donc ,contourner les sécurités Java. La sécurité n'est pas pour autant compromise, car elle est garantie par le noyau Linux.

Le navigateur Web comme le bureau utilisent les mêmes mécanismes que les autres applications. Il n'y a pas d'applications privilégiées, sauf celles présentes dans le répertoire system, qui peuvent bénéficier de privilèges particulier.

Il est possible d'installer une application à partir d'un site web, mais toujours après l'accord de l'utilisateur.

Android propose de nombreux mécanismes pour permettre la communication entre les applications. Cela permet d'ajouter de nouvelles fonctionnalités, de proposer des extensions, etc. C'est une caractéristique propre à Android, que l'on ne retrouve pas chez les concurrents. Ainsi, Android propose une plus grande intégration entre les applications. Cela présente également un challenge plus important concernant la sécurité.

Il est également possible de communiquer via des canaux caché comme la consommation CPU, la modification de l'état d'une led, etc. Mais cela est présent pour tous les OS.

Android propose à ce jour le plus grand nombre de technologies de communications avec l’extérieur : Wifi, Wifi direct, Bluetooth, Ethernet, NFC, USB, 4G/3G/2G/GSM, etc.

Publication

Pour pouvoir publier une application sur Google Market, le développement doit s’acquitter du paiement d'un droit d'accès unique de 25$. Ainsi, le certificat qu'il utilisera pour signer ces applications sera connu par Google avant de publier les applications. Cela permet de lier un utilisateur avec la carte de paiement. En cas de problème, Google peut se retourner vers le

propriétaire de la carte bancaire ayant effectuée le paiement initial.

Est-ce que cela sera toujours valide dans dix ans, lorsque les archives des propriétaires des cartes bancaires ne seront plus disponible ? Aucune information n'est disponible à ce jour.

Le certificat interdit la compromission du code. Un virus ne peut modifier la partie java d'une application, car cela invalide la signature numérique. L'application ne peut plus être exécutée.

Les librairies partagées ne sont pas signées électroniquement.

Par défaut, il n'est pas possible d'installer une application n'étant pas présente dans la place de marché. Un drapeau permet néanmoins de lever cette restriction pour pouvoir installer toutes les applications voulues, à partir d'un téléchargement de site, via le câble USB, etc.

Depuis peu, Google pratique une vérification technique sur les applications publiées. Il permet également d'identifier des applications « volée » à un développeur légitime, avant publication. Cela arrive lorsqu'un pirate reprend une application réelle, la modifie et la publie sous un autre nom.

Il y a peu d'information sur ce mécanisme de vérification. Il semble qu'une simulation d'exécution soit pratiqué dans une machine virtuelle.

Ce processus étant automatique, il est facile de corriger une faille de sécurité et de publier une nouvelle version sur la place de marché dans la minute. Il n'est pas besoin d'attendre une validation par Google.

Par contre, il est également possible de publier une application malveillante, contournant les vérifications effectuées par Google. Lorsque cette dernière sera détectée, le développeur associé sera bannis de tous les services de Google. Enregistré sur une liste noire, il ne pourra plus utiliser les services proposés. Même s'il possède plusieurs comptes, pour peu qu'a un moment donné ils ont été utilisés simultanément, tous les comptes seront banni.

Google se réserve le droit de désinstaller à distance les applications malveillantes. Il est également capable d'installer une nouvelle application à l'insu de l'utilisateur.

Chiffrement

Depuis la version 3, il est possible de chiffrer l'intégralité du disque, avec une clef dérivée du mot de passe d’allumage du téléphone. Seule une attaque par force brute permet alors de récupérer les données.

Les fichiers ne sont pas chiffrés par ailleurs. Depuis la version 4, il existe un conteneur de sécurité chiffré à l'aide de la clef de déverrouillage du téléphone.

Vulnérabilité

Plusieurs vulnérabilités ont été découvertes sous Android, essentiellement dans les composants applicatifs s'exécutant sous l'utilisateur root. Ces vulnérabilités permettent une prise de contrôle du téléphone par une application.

Par exemple, la fonction setuid() de Linux à besoin de créer un nouveau thread avant de changer le propriétaire. S'il n'y a plus de thread disponible, car le maximum est atteint, cette fonction retourne une erreur et ne fait rien. Une attaque d'Android à consisté à saturer les threads, pour générer une erreur sur cette fonction et ainsi, rester root.

Depuis la version 4, Android utilise plusieurs mécanismes pour se protéger contre l'exploitation de bugs dans les processus root :

- Un adressage mémoire aléatoire, pour réduire plusieurs familles d'attaques (ASLR) ;
- Une protection contre les débordements de tampons dans la pile (Stack-Smashing Protector[1]) ;
- Contre les manipulations d'entier (safe_iop[2]) ;
- Les allocateurs mémoires de OpenBSD pour résisteter à un double free(), à l'agrégation de chunk ou le dépassement d'entier dans un calloc().
- Utilisation de mmap_min_addr[3] pour éviter l'exploitation d'un pointeur NULL ;
- la pile et la mémoire ne sont pas exécutables.

De plus, Stephen Smalley de la NSA propose de renforcer[4] le modèle de sécurité, afin de limiter les attaques aux vulnérabilités noyaux. Il propose un patch de sécurité adapté à Android. A ce jour, nous ne savons pas si sa proposition sera présente dans les prochaines versions de l'OS.

D'autres pays travaillent à renforcer la sécurité de l'OS car c'est l'OS privilégié pour les terminaux embarqués par les militaires.

L'armé américaine à présenter des téléphones Android[5] renforcé au niveau sécurité, pour les distribuer aux militaires et ambassades.

Modèle de sécurité iPhone

Voici le modèle de sécurité proposé par Apple.

Isolation

IPhone utilise un modèle de sécurité basé sur un bac-à-sable[6]. Chaque application est isolée du système et des autres applications. Elles doivent utiliser des API pour accéder aux différents éléments du système. Cela permet de vérifier les privilèges et limite les attaques entre les applications.

Les applications tournent sous le même utilisateur : mobile. La pile et la mémoire de travail ne sont pas exécutable. Les pages exécutables ne peuvent être écrites (W^X). Depuis la version 4.3, l'exécution des applications s'effectue à des adresses aléatoires (ASLR).

Chaque application possède un répertoire de travail privé.

En plus des applications classiques, iOS supporte un nombre limité de fonctionnalité en tâche de fond pour la diffusion audio, la voie sur IP, la géolocalisation, les mécanismes de push, les notifications locale et pour terminer une tâche que l'utilisateur a démarré[7].

Il y a des applications dites « classiques », et des applications que seul Apple peut développer.

Les applications ne peuvent lancer de traitement en tâche fond, hors du contrôle du framework (fork() échoue).

La communication entre les applications iPhone est limitée. Une application peut s'enregistrer pour recevoir des notifications[8] ; elle peut répondre à un format d'URL ; elle peut communiquer via des sockets.

---

[1] http://fr.wikipedia.org/wiki/Stack-Smashing_Protector
[2] http://code.google.com/p/safe-iop/
[3] http://wiki.debian.org/mmap_min_addr
[4] http://www.h-online.com/open/news/item/NSA-releases-security-enhanced-Android-1414017.html
[5] http://edition.cnn.com/2012/02/03/tech/mobile/government-android-phones/index.html
[6] http://iphonedevwiki.net/index.php/Seatbelt
[7] http://markn.ca/2011/ios4/#high-level-design
[8] https://developer.apple.com/library/ios/#documentation/Cocoa/Reference/Foundation/Classes/NSNotificationCenter_Class/Reference/Reference.html

Depuis la version 4, il est également possible de partager des fichiers entres applications, via une copie complète de ces derniers par le système, d'une application vers une autre.

Apple ne propose pas de liste de privilège associés aux applications. L'entreprise privilégie l'approche « on demande ». Lorsqu'une fonctionnalité sensible est utilisé, une confirmation de l'utilisateur est demandé. Par exemple, récupérer les informations de géolocalisation exige une confirmation de l'utilisateur.

Ces privilèges sont finalement peu nombreux. Les applications peuvent alors exploiter toutes informations à dispositions, sans en informer l'utilisateur.

Apple indique seulement maintenant, en mars 2012, qu'une prochaine version de l'OS demandera l'accord de l'utilisateur pour permettre aux applications d'avoir accès au carnet d'adresse[9] !

Publication

Pour être publié sur la place de marché d'Apple, un développeur doit s'acquitter de 100$ par an. Les applications proposées sont validées par Apple au niveau fonctionnalité, respect des chartes de designs, utilisations des APIs, etc. avant d'être disponibles aux utilisateurs. Cela permet de garantir une certaine qualité dans les applications proposées et une cohérence dans les usages.

Les vérifications effectuées par Apple peuvent être facilement contournée par les développeurs. Quelques exemples d'applications malveillantes ont été découverts.

Apple peut rapidement supprimer ces applications et les supprimer des différents téléphones.

En cas de vulnérabilité, il est possible de demander une publication en urgence d'une application.

Pour protéger son écosystème, Apple met en œuvre de nombreuses techniques pour interdire l'installation d'applications non validées. Mais, aucune n'a résisté aux hackers. Chaque modèle peut être modifié pour supprimer les chaînes de vérifications imposées par Apple. Ainsi, l'utilisateur peut installer toutes les applications qu'il souhaite.

Mais il faut reconnaître que la sécurité globale est de plus en plus forte. Il faut réussir cinq exploits pour obtenir un jailbreak résistant au redémarrage du téléphone[10].

Chiffrement

Un composant électronique en charge des algorithmes de chiffrement est présent dans les terminaux. Ce dernier est fortement sollicité pour la lecture ou l'écriture des données sur disque. Plusieurs clefs sont utilisées, correspondant à des usages différents. Elles sont stockées dans le bloc 1 de la mémoire flash. Le composant ne peut servir de conteneur de certificat.

Lors d'une remise à zéro du terminal, le système se contente d'effacer toutes les clefs.

La clef EMF! permet de chiffrer tous le disque, même le journal HFS.

Une clef master est déchiffrée lors du démarrage du terminal, avec le code de déblocage du téléphone de l'utilisateur. Elle est gardée en mémoire en permanence. Elle permet au système de déchiffrer les informations du disque, indispensable au démarrage du système, avant l'identification de l'utilisateur. Cela permet aux applications en tâche de fond de fonctionner. Seul une attaque en force brute permet de retrouver le code de l'utilisateur pour pouvoir

---

[9] http://www.ipodnn.com/articles/12/02/15/company.responds.to.path.other.controversies/

[10] http://www.macgeneration.com/unes/voir/131112/securite-ios-a-pris-une-grosse-avance-sur-android

déchiffrer cette clef et avoir accès à tous les fichiers non protégés par le code pin de l'utilisateur.

La clef master est mémorisé dans un conteneur chiffré appelé keybag. D'autres clefs peuvent s'y trouver pour les différentes applications. Ce conteneur est lui-même chiffré à l'aide de la clef BAG1 présent dans le bloque 1 de la mémoire flash.

Une autre clef n'est présente en mémoire que lorsque l'utilisateur à débloqué son téléphone à l'aide d'un code numérique ou alpha-numérique. La clef est automatiquement effacée de la mémoire après une période d'inactivité. Les données chiffrées avec cette clef ne sont disponibles aux applications qu'après l'authentification de l'utilisateur.

Une dernière clef est générique (DKey), et est utilisé pour tous les autres fichiers. Cette dernière peut être facilement déduite des informations physiques du téléphone.

De plus, depuis la version 4, chaque fichier utilise une clef unique, associé à son nom, pour le chiffrement des données. Ainsi, pour effacer un fichier, il suffit de détruire cette clef. Cette approche permet un effacement instantané du fichier, quelque soit sa taille. Cela interdit la récupération du contenu des fichiers effacés. Mais comme la clef se retrouve parfois dans le journal du gestionnaire de fichier HFS, il est possible de récupérer les derniers fichiers effacés.

ITunes ne peut accéder qu'aux fichiers présents dans le répertoire « Media ».

Sortie de la boite, toutes les données utilisateurs utilisent le chiffrement générique. Seules les emails et les pièces jointes utilisent le chiffrement associé à la session de l'utilisateur.

En accédant au bloque 1 de la mémoire flash, il est possible de récupérer une très grande partie des informations présentes sur le disque. Malheureusement, plusieurs vulnérabilités permettent de récupérer une grande partie de ces informations. Pour les fichiers restants, une attaque en force brute sur les 10.000 combinaisons peut être pratiquée en une vingtaine de minutes.

Le vol ou l'emprunt d'un téléphone permet de rapidement récupérer toutes les informations confidentielles. Le mode de boot DFU (« Device Firmewark Update ») permet de démarrer le terminal dans un mode spécial, contournant toutes les sécurités mises en place par l'OS. Il est alors possible d'accéder aux différentes clefs, d'injecter des modifications à l'OS, de récupérer une copie de l'intégralité du disque, etc.

Il est donc important d'utiliser un mot de passe long et complexe pour protéger en partie le terminal.

Même si plusieurs processus sont exécuté sous le même utilisateur, elles n'ont pas pour autant les mêmes privilèges. Certaines ont des privilèges spécifiques, définies dans un profil de provisioning, signé par Apple. Des apis critiques vérifient la signature des applications avant d'exécuter les traitements critiques.

Vulnérabilité

Aucune version n'a résisté aux attaques des hackers. Il y a toujours une technique permettant de modifier le système pour alléger les contraintes imposées par le système. Cela permet l'installation d'applications non validé par Apple.

Modèle de sécurité Windows Phone 7

Et enfin, le modèle de sécurité proposé par Microsoft.

Isolation

Windows Phone utilise différentes « chambre[11] » isolant les différents privilèges. Il y a quatre type de chambre. Trois ont des privilèges fixés :

[11] http://www.microsoft.com/download/en/details.aspx?id=27743

- Le Trusted Computing Base (TCB) possède le maximum de privilèges. Les traitements ont un accès complet à l'intégralité du système. Le noyau du système et les drivers s'exécute dans cette chambre. Réduire au minimum les traitements à ce niveau permet de limiter les risques de sécurités.
- Le Elevated Rights Chamber (ERC) permet d'avoir accès à toutes les ressources, excepté les politiques de sécurités. Ce niveau est destiné aux services et aux drivers de niveau « utilisateur », partagés par les différentes applications. Seul Microsoft peut ajouter des services pour ce niveau de privilège.
- Le Standard Rights Chamber (SRC) est le niveau de privilège pour les applications pre-installées sur la plate-forme (Microsoft Outlook Mobile, etc.). Elles ne peuvent proposer de services.
- Le Least Privileged Chamber (LPC) est le niveau où sont installé les applications qui ne viennent pas de Microsoft. Ce niveau est ajusté suivant les privilèges nécessaires à chaque application.

Windows Phone impose un seul langage de développement pour Windows Phone : .Net managé.

Ce langage est complètement contrôlé par la machine virtuelle. Le typage fort, la vérification des limites et les fonctions de gestion de la mémoire du code managé peut aider à éliminer ou à minimiser les nombreuses erreurs de programmation courantes qui peuvent conduire à l'exploitation d'applications par des pirates ou des hackers.

Ainsi, il n'est pas possible d'en sortir pour exécuter du code natif. Cela améliore la sécurité. Des outils permettent d'identifier des approches de codage pouvant présenter des risques. Ils sont appliqués à toutes les applications publiées sur la place de marché de Microsoft.

Une seule application est active à la fois et il n'est pas possible d'avoir des traitements en arrière plan. Lorsqu'une nouvelle application est présenté à l'utilisateur, la précédente est tuée. Une pile d'activité permet de relancer une ancienne application lors de l'extinction de l'application principale, lors de l'appuie sur le bouton « retour ».

Le multitâche n'est pas possible, ce qui interdit des traitements cachés ou des espions logiciels. Il est interdit de communiquer directement entre les applications. Il faut utiliser un serveur intermédiaire sur le réseau. L'exposition aux risques entre les applications est alors limité. Il est alors impossible de faire fonctionner la plupart des applications en absence du réseau.

Les applications peuvent souscrire à des agents fonctionnant en tâche de fond, comme le transfert de fichier mais ne peuvent pas exécuter des applications en tâche de fond. Quand un utilisateur change d'application, l'application précédemment utilisée est endormie. Cette conception permet d'éviter que des applications utilisent des ressources critiques.

Les applications ne peuvent accéder directement aux périphériques du terminal ou à la mémoire des autres applications, y compris le cache du clavier. Elles doivent passer par des API spécifique en .NET.

Les applications peuvent exiger quelques « capabilities »[12]. (les informations de localisation géographique, caméra, microphone, le réseau et les capteurs). Il n'est pas possible d'augmenter la liste des privilèges.

Ces informations doivent être déclarées dans le fichier WMAppManifest.xml par le développeur.

Lors de l'installation de l'application, Windows Phone prépare un espace isolé dont les privilèges sont limités à ceux déclarés. Ces informations sont présentées à l'utilisateur sur la place de marche, lors de l'installation et avant d'être utilisées par l'application.

Cette approche permet de réduire la surface d'attaque des applications.

[12]http://blogs.msdn.com/b/jaimer/archive/2010/04/30/windows-phone-capabilities-security-model.aspx

Cette approche est pratique pour les développeurs, mais ne permet pas d'ajouter de nouveaux privilèges. Comme les applications ne peuvent communiquer entre-elles, cela n'est pas gênant.

Le navigateur Internet n'utilise pas de privilèges particulier et interdit l'utilisation de composants sensibles comme les extensions. Il n'est pas possible d'installer une application depuis le navigateur.

Publication

Les développeurs doivent s'acquitter d'un droit annuel de 100$ pour pouvoir publier une application. Des vérifications sont effectués par Microsoft avant d'apposer sa signature numérique, permettant l'exécution de l'application dans le téléphone.

Les privilèges nécessaires à l'application sont présentés à l'utilisateur au plus tôt, avant l'installation de l'application.

Des applications non-officielles ont été publiés et validées (deezer avec un 'd' minuscule) démontrant la fragilité de toutes les places de marchés. Rien ne permet de garantir qu'une application est associée aux sites qu'elle utilise.

Chiffrement

Chaque application possède un répertoire de travail, inaccessibles aux autres applications. L'accès direct aux fichiers est impossible. Il est nécessaire d'utiliser un objet spécifique pour cela.

Lorsqu'une carte SD est présente, le système de fichier est lié à la carte interne. Par mesure de protection, retirer la carte SD réinitialise les données utilisateurs restées sur la mémoire interne du téléphone, à moins que la carte SD soit réinsérées. L'insertion d'une autre carte SD efface et réinitialise le téléphone avec la nouvelle carte.

Les Windows Phone mettent en œuvre un mécanisme de verrouillage de la carte SD standard. La carte utilise une clé unique de 128 bits spécifique du téléphone. L'accès à la carte SD est empêchée par le contrôleur de carte SD sans fournir le bon mot de passe de 128-bits, ce qui rend extrêmement difficile d'accéder aux données d'une carte SD orpheline.

La synchronisation des données sensibles (mail, fichiers) ne peut s'effectuer par une connexion USB. Seul des connexions réseaux chiffrées sont possibles. La synchronisation par USB est possible pour les médias Image, son et vidéos.

Vulnérabilité

Il existe quelques vulnérabilités permettant de débloquer un téléphone Windows Phones. Des vulnérabilités permettent d'importer des DLL native dans une application[13].

Modèle de sécurité Windows Phone 8

Les détails du modèle de sécurité de Windows Phone 8[14] ne sont pas connu. Néanmoins, nous savons qu'il y a une remise en cause complète des choix de la version 7. Il sera possible de rédiger des applications en code natif, de communiquer entre les applications, d'utiliser une carte mémoire SD, etc.

Windows Phone 8 n'est pas une évolution de Windows Phone 7, mais un nouvel OS.

---

[13] http://www.schuba.fh-aachen.de/papers/11-ICDF2C.pdf

[14] http://pocketnow.com/windows-phone/exclusive-windows-phone-8-detailed

Conclusions

Nous constatons que les trois OS utilisent des approchent parfois similaires, parfois très différentes. Android est le système le plus ouvert et le plus souple ; IOS privilégie la protection du modèle économique d'Apple ; Microsoft utilise une approche tellement faible que tous sera revu pour la prochaine version.

Philippe PRADOS article@prados.fr
Architecte Senior Smart Mobility - AtoS