Tags NFC - getting started

Le projet EsupNfcTag a pour but d'intégrer l'usage des cartes sans contacts dans les applications métiers du système d'information de l'établissement.

EsupNfcTag supporte notamment les possibilités se sécurité des cartes à puce sans contact Mifare DESFire EV1.

Cette page tente ici de retranscrire l'étude technique qu'il a fallu mener pour arriver à rendre EsupNfcTag fonctionnel - notamment autour de la partie Mifare Desfire.

Elle peut ainsi constituer une documentation type "getting starting" pour le technicien souhaitant manipuler la carte multi-services (Mifare Desfire) de son établissement.

Cette documentation vise donc plutôt à donner les outils permettant de manipuler la carte directement et simplement.

Identification

La carte à puce étudiant/professionnel/campus d'un établissement a vocation à être une carte multi-services. Son objectif est ainsi de proposer un panel de services intra ou extra établissement pour leurs détenteurs.

Dans les faits elle ne propose alors usuellement que des services d'authentification/identification (sauf exception si votre carte fait office de carte pour les transports en commun par exemple ...) notamment pour :

- s'identifier sur des bornes d'accès pour ouvrir des portes
- s'identifier et ainsi “payer” via son compte “Izly” dans les restaurants du CROUS
- s'identifier sur des copieurs pour “payer” des photocopies, retirer des impressions ou encore scaner
- ...

2 types d'authentification/identification sont alors mis en oeuvres :

CSN / UID

L'identification par CSN est la plus simple à mettre en oeuvre. Véritablement issue d'une norme supportée par l'ensemble des cartes actuelles, la récupération du numéro de série est effectivement standardisée.

La commande APDU PC/SC suivante (5 octets en hexa) FFCA000000 permet en effet de récupérer ce CSN / UID sur le bloc0. On récupère cette information en clair (sans authentification / communication chiffrée) et donc tout lecteur peut lire également cette information.

L'écriture de l'UID sur le bloc0 est normalement verrouillée sur les cartes proposées par les distributeurs officiels - cependant n'importe qui pouvant acheter des “magic cards” chinoises pour quelques (centimes d') euros, le piratage de ces cartes est un alors un “jeu d'enfant” … modulo qu'on ait le bon outil et la bonne commande pour ce faire.

"Application" spécifique

Les cartes (à puce ou sans contact - on parle souvent de “tag”) proposent un système de fichiers “assez” standardisé. On y retrouve ainsi des répertoires qui contiennent eux-mêmes des fichiers. Les cartes mifare desfire proposent la possibilité de stocker des applications (~dossiers) qui elles même peuvent stocker des fichiers.

La lecture des fichiers détenus par une application peut se faire sans authentification ou ne peut être disponible que si l'on fournit une clef d'authentification spécifique à l'application (ou plutôt même au fichier lui-même).

C'est ce qui peut permettre de véritablement sécuriser ici l'authentification/identification de la carte.

Aussi une carte peut contenir plusieurs applications comme une application spécifique à un contrôle d'accès, l' application “crous” pour les restaurants universitaires, ... ou encore une ou plusieurs "applications" institutionnelles.

Les applications d'une même carte ont chacunes leurs propres clefs, et les établissements eux-mêmes n'ont pas forcément toutes les clefs de toutes les applications qu'ils disposent sur leur propre carte.

Outillage sous Android

Les applications Android sont parfaites pour se faire une idée de ce que vous avez dans vos tags.

TagInfo de NXP permet depuis son téléphone Android d'avoir un maximum d'informations sur un tag NFC juste en badgeant votre carte.

Mifare Desfire Tool vous permet quant à lui d'aller jusqu'à vous authentifier et déchiffrer un fichier d'une application Desfire ... évidemment, il vous faut alors lui fournir la clef.

Ensuite Android et Java fournissent également une bonne API pour lancer des commandes à la carte à puce ; c'est ce qu'utilise EsupNfcTagDroid.

Outillage sous Windows

ReadCard permet d'afficher la structure de la carte et de voir certaines information (CSN, etc... voir plus si on renseigne les clé de lecture)

Outillage sous linux

PCSC (libpcsclite) est une api/bibliothèque permettant d'accéder aux cartes à puce.

Elle permet de passer des commandes à une carte … encore faut-il savoir quelles commandes passer !

Pour identifier ces commandes (+/- spécifique à chaque technologie de carte), il est intéressant de manipuler les cartes via PCSC (depuis spector par exemple) ou encore via libnfc.

PCSC supporte bon nombre de lecteurs nativement (avec les drivers open-source livrés avec) ou après installation du driver propriétaire.

Pour le lecteur “Identive cloud 4700 f” par exemple, il a fallu installer le driver propriétaire suivant pour que le lecteur sans contact fonctionne (en plus du lecteur avec contact) :

http://support.identive-group.com/download/driver/scmccid_5.0.33_linux_32bit_rel.zip

Aussi, concernant les lecteurs de cartes, on préférera des lecteurs qui permettent une mise en place facile et native de ces outils : PCSC *et* libnfc. libnfc n'est supporté que par très peu de lecteurs … avant d'acheter un lecteur, il vaut donc mieux regarder quels sont les lecteurs reconnus par la communauté comme compatibles. http://nfc-tools.org/index.php?title=Devices_compatibility_matrix

3 lecteurs / clefs usb qui nous paraissent ainsi intéressants par exemple :

- SCL3711 (germany)
- StickID (italy)
- DDS PN533 (india)

Si on souhaite se réaliser un petit gadget indépendant (avec arduino par exemple), la puce “PN532 NFC/RFID” est intéressante : https://www.itead.cc/wiki/ITEAD_PN532_NFC_MODULE ou encore http://www.adafruit.com/products/364

Enfin notons qu'un outil qui semble fort apprécié des bidouilleurs/hackeurs en tout genre est le **proxmark3** …

Tests (**sous linux ici**)

Dans les faits, si vous souhaitez simplement consulter / manipuler / lire votre carte rapidement, il est plus simple d'utiliser directement une application Android toute faite.

pcsc_scan

Cette commande permet de récupérer la liste des lecteurs PC/SC disponibles (notamment dans le cadre où notre lecteur est à la fois avec et sans contact, il est vu comme 2 lecteurs …).

Cette commande nous permet également de valider notre installation (libpcsclite, pcsc-tools, drivers propriétaires éventuels …)

scriptor ou opensc-tool

Ces outils nous permettent de passer des commandes via PCSC.

Pour lancer scriptor (il faut positionner la carte avant sur le lecteur) : scriptor -r “CLOUD 4700 F Smart Card Reader [CLOUD 4700 F Contactless Reader] (53201322200637) 01 00”

FF CA 00 00 00 nous renvoie par exemple 04 53 71 D2 FD 3A 80 90 00

04 53 71 D2 FD 3A 80 est le CSN - 90 00 correspond au code de retour (opération ok).

90 6A 00 00 00 permet de lister les applications - il renvoie par exemple ici C0 85 F5 00 84 F4 40 85 F5 C1 85 F5 91 00.

En enlevant le code retour et sachant que les identifiants des applications sont sur 3 octets, nous retrouvons l'application “C0 85 F5”, “00 84 F4”, “40 85 F5” et “C1 85 F5”.

Grâce à la liste des applications enregistrées sur NXP nous arrivons finalement à retrouver à partir de ces identifiants les noms des applications.

En Desfire, les réponses se lisent de droite à gauche 2 bytes par 2 (cf 'LSB') :

“C0 85 F5” devient “F5 85 C0” - on retrouve alors 585C le code pour “identification for academic services” de “Normandie Universite”

“00 84 F4” devient “F4 84 00” - on retrouve alors 4840 le code pour “access contr,time&attendance,multifunccard” de “Horoquartz”

“40 85 F5” devient “F58540” - on retrouve alors 5854 le code pour “identification of students inench Universities (unique student number among all the country)” du “CNOUS”

“C1 85 F5” est la dernière application en date (“dérivée” de 585C) et est l'application demandée par la comue qui présente l'énorme intérêt que l'"on" dispose des clefs pour lire les fichiers présents. Ces fichiers présentent toutes un identifiant spécifique que l'on connait dans nos bases chiffrées avec 3 clefs différentes (pour des questions de confort).

On sélectionne l'application “C1 85 F5” ainsi : 90 5A 00 00 03 C1 85 F5 00

On liste les fichiers en passant la commande 90 6F 00 00 00 qui nous renvoie 00 01 02 91 00.

Le code retour 91 00 en moins, on obtient 3 octets représentant les identifiants des fichiers : 00, 01 et 02

On peut encore passer différentes commandes, notamment pour connaître les “file settings” de chaque fichier : 90 F5 00 00 01 00 00 nous renvoie 00 00 44 14 1F 00 00 91 00 - le deuxième 00 nous renseigne notamment sur le fait que le mode de communication est en plain text (en clair), cad non encrypté … [ce qui ne doit usuellement pas être le cas sur les applications sécurisées en fait , sauf pour test].

Il faut ensuite s'authentifier en utilisant la clef spécifique au fichier que l'on souhaite lire.

C'est une authentification AES - pas des plus simples à mettre en oeuvre - et qui nécessite alors un peu d'algorithmique - celui-ci est bien expliqué sur ce document http://www.ti.com/lit/an/sloa213/sloa213.pdf à la page 6 précisément.

En suivant http://stackoverflow.com/questions/21257442/mifare-desfire-ev1-authentication-using-aes (nous permet notamment de savoir quoi mettre en tant que “IV”) on arrive à s'en sortir et il est possible de jouer l'ensemble de la procédure d'authentification en lignes de commandes (en s'aidant d'openssl pour décrypter/encrypter les différents nombres reçus et à envoyer - exemple :

```
echo bf7e381e6ae0c91abcdecac82abee3b2 |xxd -p -r|openssl enc -aes-128-ecb -d -K AE17129EA5C4D303F025032E2FCA2CC6 -
iv 00000000000000000000000000000000 -nopad |xxd -
```

Nous l'avons fait de notre côté en Java (d'auant que du code est librement disponible pour ce faire, voir le code utilisé dans EsupNcTag par exemple).

Une fois l'authentification réussie, on peut enfin lire le fichier voulu : la commande suivante 90BD000007000000016000000 permet de lire les 22 premiers octets du fichier 00. Fichier qu'il faut ensuite éventuellement déchiffrer.

cardpeek

Utilitaire graphique qui permet d'afficher les informations détenues par une carte en utilisant PCSC. Des scripts sont donnés par défaut pour un certain nombre de type de cartes. On a pu lire avec succès les cartes Astuce et Vital par exemple : les données sont en claires et ne nécessitent pas de clef d'authentification.

libnfc

Différent de PCSC, libnfc utilise NFC comme protocole de communication pour interagir avec la carte.

Propose de base, via le package libnfc-examples, des utilitaires plus avancées que ce que propose pcsc-tools. On y trouve notamment nfc-mfsetuid qui permet en une commande de modifier le CSN d'une carte (d'une magic-carte dont le bloc 0 n'est pas verrouillé).

Pour pouvoir par exemple facilement modifier le CSN d'une magic-card, il faudrait un lecteur comme la clef usb “Identive SCM SCL3711”.

Bibliographie

Quelques articles/pages précieuses :

MISC 52 : article “lecture d'une carte sim avec pssi”

Ridrix's Blog : https://ridrix.wordpress.com/category/security/smartcard-security/

NEW : projet très bien documenté d'ouverture de porte avec un Arduino et des cartes NFC Desfire EV1 (propose de plus une librairie arduino Desfire EV1 complète) - dès maintenant une référence : http://www.codeproject.com/Articles/1096861/DIY-electronic-RFID-Door-Lock-with-Battery-Backup

Liste des applications enregistrées : http://www.nxp.com/documents/other/MAD_list_of_registrations.pdf (*lien cassé, il semble que cette liste ne soit plus rendue publiquement disponible par NXP*) - on y retrouve Normandie Universite et le CNOUS.

Explication de MIFARE Secure Access Modules (SAM AV2) - a priori utilisé par Izly pour écrire / coder l'application CNOUS (CROUS) dans les cartes NFC via l'usage supplémentaire d'une carte sim (avec clef usb faisant office de lecteur) : https://github.com/islog/liblogicalaccess/wiki/MIFARE-SAM-AV1-AV2-examples

A lire … :

MISC HS 2 : Cartes à puce - Découvrez leurs fonctionnalités et leurs limites

Plein de ressources/logiciels : http://wiki.yobi.be/wiki/RFID