SERMA TECHNOLOGIES

Serma Group

Compromission de carte à puce via la couche protocolaire ISO 7816-3

Symposium sur la sécurité des technologies de l'information et des communications, 3 au 5 juin 2015

Guillaume VINET

SERMA TECHNOLOGIES
Serma Group

PLAN

- Sécurité des cartes à puces
- Pourquoi et comment attaquer la couche ISO 7816-3
- Analyse de vulnérabilité de l'ISO 7816-3
- Mise en place des attaques
- Résultats
- Conclusion et perspective

Carte à puce et Sécurité

De nombreux cas d'utilisation …

Santé (Vitale)

Carte IAS (identification, authentification et signature).

moneo
PAYMENT SOLUTIONS

Porte-monnaie électronique

Téléphonie (SIM)

Cartes bancaires

… et de nombreux schémas certifiant leur sécurité :

VISA

VISA Chip Security Program (VCSP)

Common Criteria

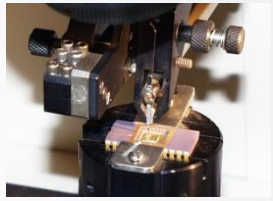
Critères Communs (ISO 15408)

MasterCard
Worldwide

MasterCard Worldwide Compliance Assessment and Security Testing program (CAST)

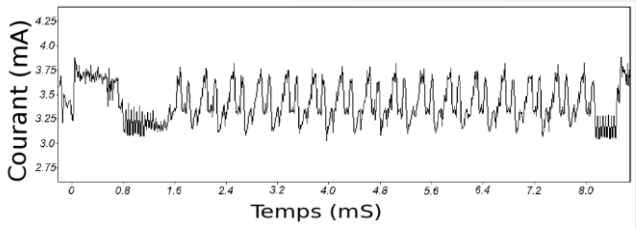
Types d'attaques couverts

Types d'attaque couverts :



- Attaque semi-invasive par faute : injection laser ou électro-magnétique,

- Attaques par canaux cachés : Single Power Analysis (SPA), Differential Power Analysis (DPA) ,



- Attaque sur plateforme Java Card : application malveillante ou attaque combinée.

Et sur la couche de communication bas-niveau : l'ISO 7816-3 ?

Etat de l'art

Etat de l'art:

- "Advisories : PCSC-Lite: pcscd ATR Handler Buffer Overflow", MWR Labs, December 13, 2010,
- "Un framework de fuzzing pour cartes à puce: application aux protocoles EMV", Julien LANCIA SSTIC 2011,
- PINPADPWN, Nils et Rafael Dominguez Vega, Black Hat 2012,
- Compromission d'un terminal sécurisé via l'interface carte à puce, Guillaume VINET, SSTIC 2013,
- Mission Mpossible, Nils et Jon Butler, Black Hat 2014.

Les attaques ciblent :

- ou bien des terminaux ou des logiciels clients,
- ou bien des cartes, mais au niveau applicatif.

➢ **Y-a-t'il un intérêt à attaquer la couche de communication bas-niveau des cartes contacts : l'ISO 7816-3?**

SERMA TECHNOLOGIES
Serma Group

Pourquoi et comment attaquer l'ISO 7816-3?

Intérêts :

- Parc des cartes contacts toujours très important. Plupart des cartes sans-contact sont *duals* (support du contact).
- Indépendant de l'application embarquée, donc applicable sur n'importe quelle carte contact.
- Cas des cartes basées sur une machine virtuelle, par exemple Java Card :
  - En cas de débordement dans un tableau, levée d'une exception « *ArrayIndexOutOfBoundsException* ».
  - Pile ISO 7816-3 toujours implémentée en code natif (C ou assembleur). A ce niveau, pas de détection du débordement par la machine virtuelle.

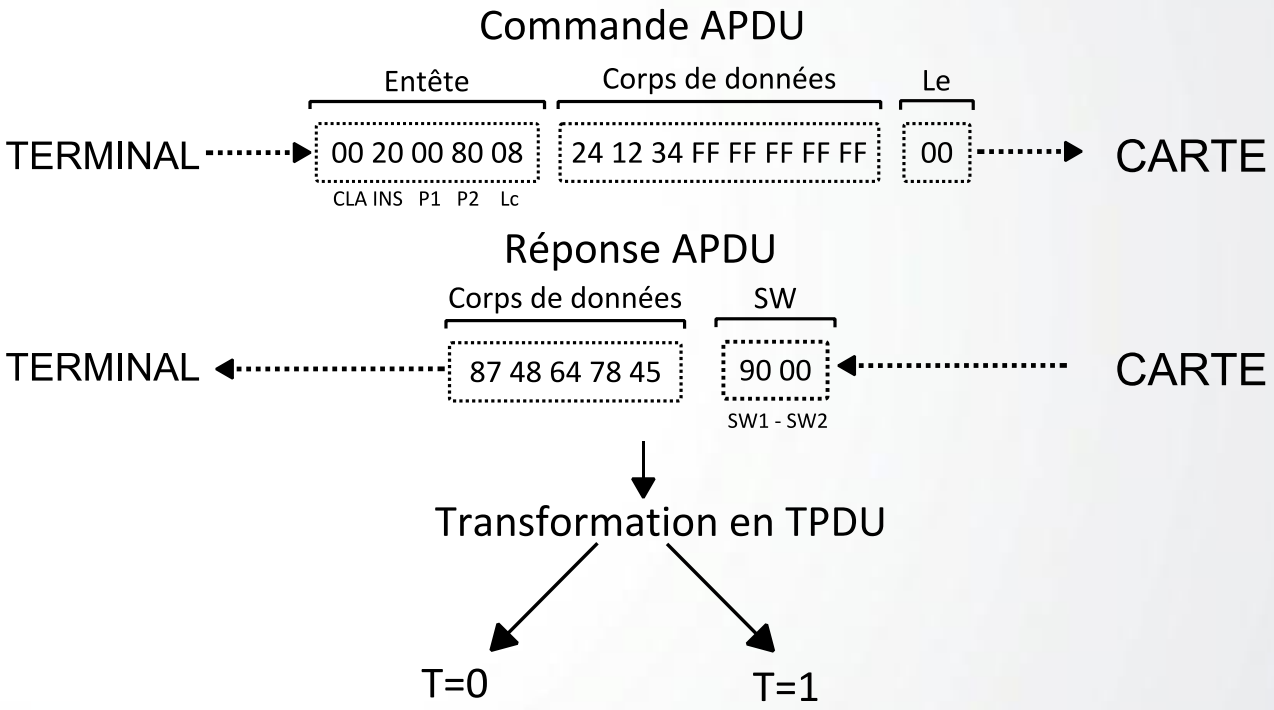
Mise en pratique grâce à des tests de fuzzing :

- Envoie de commandes correctes, mais qui ne sont pas attendues dans l'état courant.
- Envoie de commande, mais avec une structure incorrecte.

Analyse de vulnérabilité de l'ISO7816-3

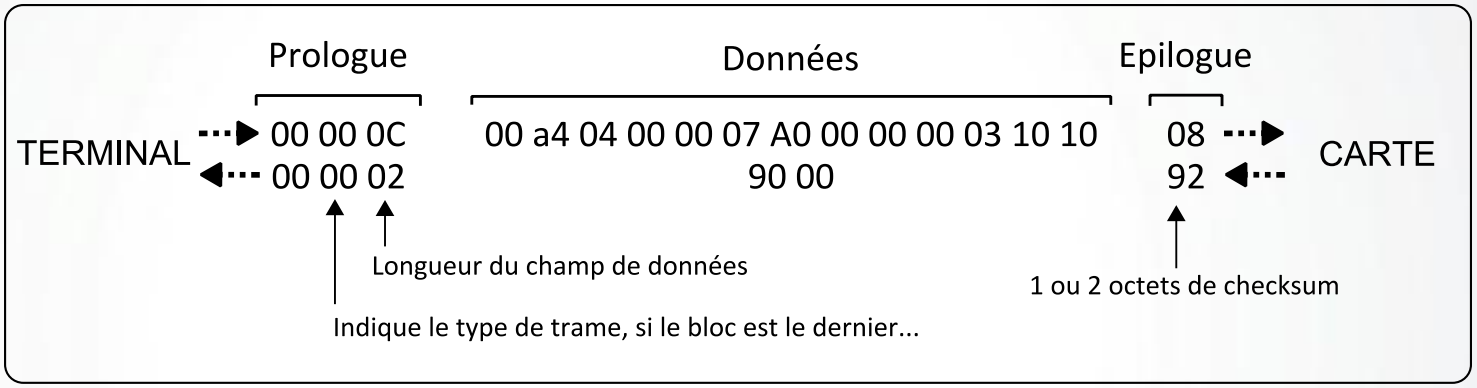
- APDU et TPDU

Couches OSI	
7.Application	APDU
6. Présentation	
5.Session	APDU
4.Transport	
3.Réseau	
2.Liaison	TPDU
1.Physique	Transmission signaux



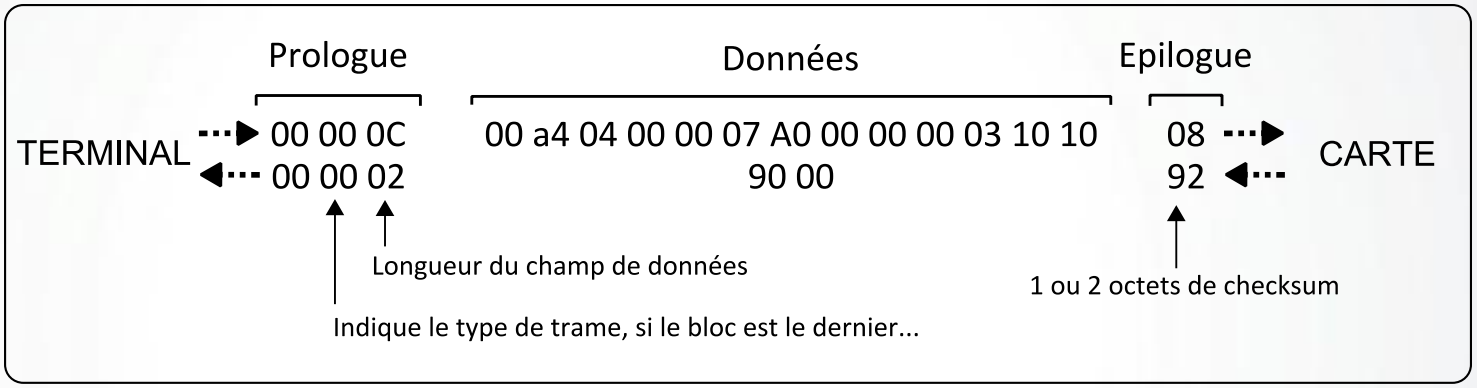
Analyse de vulnérabilité de l'ISO7816-3 : T=1

- Une encapsulation de l'APDU :



Analyse de vulnérabilité de l'ISO7816-3 : T=1

- Une encapsulation de l'APDU :



- Différents types de blocs :

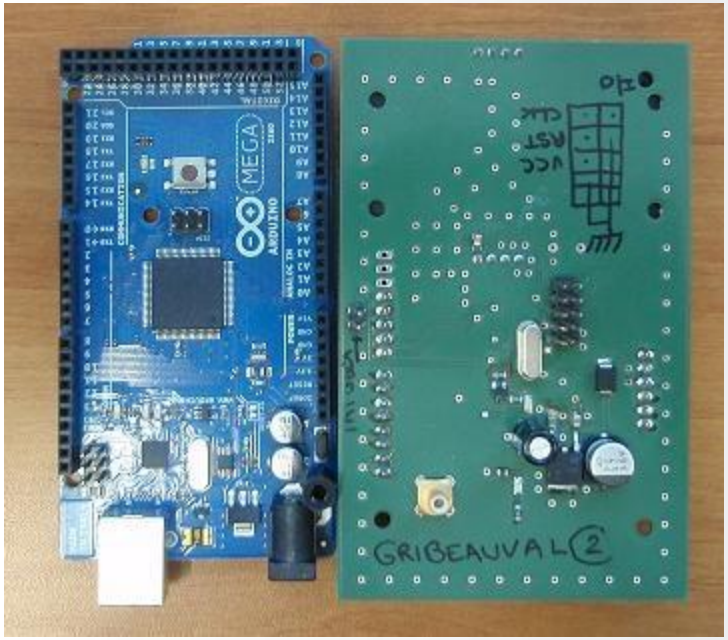
Configuration de la taille de bloc à 0x07		
00 C1 01	07	C7
00 E1 01	07	E7

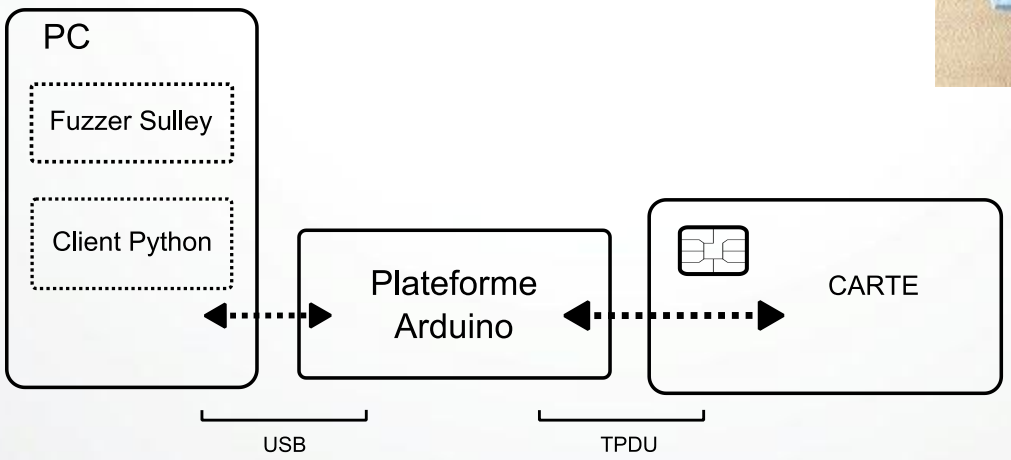
Sélection de l'application et 1er bloc de réponse de la carte		
00 00 05	00 a4 04 00 00	A5
00 20 07	6f 10 84 08 a0 00 00	74

2ème bloc de réponse de la carte		
00 90 00		90
00 60 07	01 51 00 00 00 a5 04	96

Mise en place des tests

- Notre solution
  - Implémentation d'un lecteur à base d'une plateforme Arduino et d'une extension (*shield*) propriétaire.
  - Pilotage par le PC via un logiciel client en python. Framework avec une implémentation complète du protocole T=0 et T=1, ainsi qu'un mode pour envoyer/recevoir des trames brutes.
  - Utilisation du framework de fuzzing Sulley disponible publiquement.





Résultats

10 cartes testées avec des configurations variées :

- Mode T=0 ou T=1,
- En Java Card ou en C/Assembleur,
- Plateforme Java Card ouverte ou fermée,
- Applications bancaires, de contrôle d'accès ou d'Identité.

Fuzzing en boite noire, puis requête du code source/fichiers issus de la compilation (si possible).

Résultats :

Identification	Langage	Protocole	Faille de sécurité
Carte 1.1	Java Card 2.2.X	T=1	accès illégal en écriture
Carte 1.2	Natif	T=1	accès illégal en écriture
Carte 1.3	Java Card 2.2.X	T=1	accès illégal en écriture
Carte 2	Natif	T=1	débordement de tampon dans un tableau global

SERMA TECHNOLOGIES
Serma Group

Débordement de tampon

```
global  byte ptr_io_buffer[255];
U8 counter;
U8 byte;
// Réception en-tête (NAD, PCB, LEN)
…

// Boucle de réception du champ de données + LRC
while(1){
    if ( READER_TIMEOUT == getIOByte(&byte) ){
       break;}
    *ptr_io_buffer++ = byte;
    lrc_checksum ^= byte;
    counter++;
}
// 254 octets de données + 1 octet de LRC
if (counter > 255){
    return ERROR;}

if (counter != Lc_field){
    return ERROR;}
```

SERMA TECHNOLOGIES
Serma Group

Débordement de tampon

```
global  byte ptr_io_buffer[255];
U8 counter;
U8 byte;
// Réception en-tête (NAD, PCB, LEN)
…

// Boucle de réception du champ de données + LRC
while(1){
    if ( READER_TIMEOUT == getIOByte(&byte) ){
       break;}
    *ptr_io_buffer++ = byte;
    lrc_checksum ^= byte;
    counter++;
}
// 254 octets de données + 1 octet de LRC
if (counter > 255){
    return ERROR;}

if (counter != Lc_field){
    return ERROR;}
```

1er problème : condition d'arrêt basée sur un timeout du lecteur. Possibilité d'envoyer autant d'octets que voulu à la carte.

Débordement de tampon

```
global  byte ptr_io_buffer[255];
U8 counter;
U8 byte;
// Réception en-tête (NAD, PCB, LEN)
…

// Boucle de réception du champ de données + LRC
while(1){
     if ( READER_TIMEOUT == getIOByte(&byte) ){
        break;}
     *ptr_io_buffer++ = byte;
     lrc_checksum ^= byte;
     counter++;
}
// 254 octets de données + 1 octet de LRC
if (counter > 255){
     return ERROR;}

if (counter != Lc_field){
     return ERROR;}
```

2ème problème : compteur d'*overflow* codé sur un octet. Possibilité de le faire déborder pour retomber sur une valeur cohérente avec le champ de longueur Lc.

Débordement de tampon

```
global  byte ptr_io_buffer[255];
U8 counter;
U8 byte;
// Réception en-tête (NAD, PCB, LEN)
…

// Boucle de réception du champ de données + LRC
while(1){
    if ( READER_TIMEOUT == getIOByte(&byte) ){
       break;}
    *ptr_io_buffer++ = byte;
    lrc_checksum ^= byte;
    counter++;
}
// 254 octets de données + 1 octet de LRC
if (counter > 255){
    return ERROR;}

if (counter != Lc_field){
    return ERROR;}
```

Exemple de trame invalide acceptée :

00 00 0C

00 A4 04 00 08 41 49 44 42 49 44 4f 4e

FF … FF (256 fois)

0C

- ✓ LRC cohérent sur la trame
- ✓ counter = 0xC + 256 = 0xC modulo 256

SERMA TECHNOLOGIES
Serma Group

Débordement de tampon

```
global  byte ptr_io_buffer[255];
U8 counter;
U8 byte;
// Réception en-tête (NAD, PCB, LEN)
…

// Boucle de réception du champ de données + LRC
while(1){
    if ( READER_TIMEOUT == getIOByte(&byte) ){
       break;}
    *ptr_io_buffer++ = byte;
    lrc_checksum ^= byte;
    counter++;
}
// 254 octets de données + 1 octet de LRC
if (counter > 255){
    return ERROR;}

if (counter != Lc_field){
    return ERROR;}
```

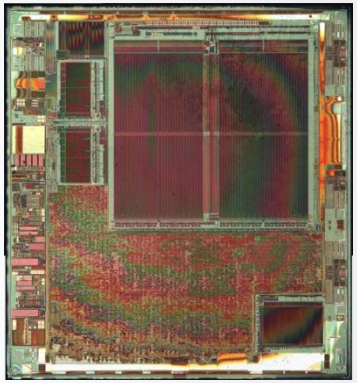
Exploitation :

Débordement dans un tableau global
Accès illégal en RAM aux structures codant les privilèges d'accès.
Modification des droits et accès illégal aux objets de l'application.

Difficultés rencontrées

Identification difficile:

- Sur les cartes implémentation de mécanismes de détection d'attaque entraînant un blocage lors de l'atteinte d'un seuil. Dans notre cas, pas de détection : possibilité de tester autant de valeurs que possibles.
- Code et fichiers de compilation (mapping mémoire) disponibles :
  - Vérification de la présence de structures intéressantes situées après le tableau d'I/O.
  - Sur les cartes, protection des structures sensibles grâce à des checksums pour se prémunir contre les attaques par faute. Connaissance de la façon de modifier les structures.



➢ Une caractérisation de l'attaque en boite noire est donc très difficile.

SERMA TECHNOLOGIES
Serma Group

Conclusion

- L'ISO 7816-3 : théoriquement intéressant à attaquer, plus particulièrement dans le mode T=1 à cause de l'encapsulation des données.

- Mise en place : réalisation d'un lecteur à base d'une plateforme Arduino pour l'envoi de commandes brutes.

- Résultat : débordement de tampon permettant un accès illégal à des biens protégés de l'application.

Perspectives

- Utilisation du lecteur/framework en combinaison avec des fuzzers applicatifs de carte à puce.

- Extension du framework aux cartes sans-contact : structure de trame proche de celle utilisée dans le protocole T=1.

- Détection de débordements dans des tableaux locaux:
  - Actuellement, espace mémoire limitée sur carte à puce, donc utilisation de tableau globaux.
  - Apparition de nouveaux microcontrôleur sur le marché avec plus de mémoire. Possibilité de changement d'implémentation avec utilisation de tableaux locaux. Type de produit à surveiller.

SERMA TECHNOLOGIES
Serma Group

Questions?

Contact : gDOTvinetATsermaDOTcom