Mission du challenge : Mission Impossible

Sommaire:

Contexte

Lors d'un examin vous n'avez pas réussie à avoir la moyenne. Cette note ne vous permet de pas de valider votre diplome et cela n'est pas envisagable pour vous. C'est pour cela que vous avez comme fabuleuse idée de récupérer le mot de passe de votre enseignant avec un keylogger pour pouvoir modifier votre note et ainsi valider l'année.

Vous ête dans l'option 2SU (option à l'INSA CVL en Sécurité et Technologie Informatique : Sécurité des Systèmes Ubiquitaire), et vous savez que votre enseignant donne des cours au 4AS (option à l'INSA CVL en Sécurité et Technologie Informatique : Architecture Administration Audit de Sécurité). Il y a une porte que cette enseignant empreinte qui relie ces deux salles.

Seulement votre enseignant à mis en place des système de surveillence pour que seulement les profféseurs puissent l'utiliser. Les systèmes mis en place sont les suivant :

- Caméra de surveillence qu'il regarde régulierement avec détection de présence
- Ouverture de la porte par carte à puce

Vous devez donc réussir à utiliser cette porte protégé par la lecteur de carte pour placer votre keylogger sans être détecté par la caméra.

Information récolté sur le système de surveillence

Pour réussir cette attaque, vous avez réussie à récolter différentes information suite à des recherche et de l'observation.

Architecture du système

En visualisant le système vous avez déterminer qu'il y a deux Raspberry PI qui communique en réseau via un switch qui est cleustorisé. La premier Raspberry à une caméra branché, et la seconde joue plusieurs rôle. Cette deuxième Raspberry indique si il y a une personne devant la caméra, ou une déconnection réseau, mais aussi permet d'intéragir avec une carte à puce qui néssécite un PIN.

Siwtch Wifi

Pour la connection au réseau Wifi, vous avez vu votre enseignant s'y connecter en tapant sont mot de passe, seulement vous ne l'avez pas exactement. Le schéma du mot de passe que vous avez déterminer est de la forme:

2SU-XX-YZ-ZZ

avec:

- X : caractère majuscule
- Y : caractère minuscule
- Z : un chiffre

Indicateur de détection de présence

Un indicateur limineux est présent qui permet de détecter la présence d'une personne devant la caméra.

- Indicateur vert : personnes n'est devant la caméra
- Indicateur Orange : Problème réseau, la caméra n'est pas joingnable en réseau
- Indicateur Rouge : Uner personne est détecté devant la caméra

Si l'indicateur est `Orange` ou `Rouge` la porte se bloque et ne peut pas être ouverte. La caméra est placé dans un emplacement statégique, on est obligé de passé devant pour accéder à la porte, et nous ne pouvons pas l'atteindre sans passer devant non plus.

Lecteur de carte

Pour pouvoir ouvrir la porte, il faut une carte à puce, l'introduire dans le lecteur de carte, puis entrer un code PIN. Vous avez réussie à récupérer le code contenue dans la carte d'un autres enseignant mais cette carte à étais désactivé donc la reproduction d'une carte avec ce code ne sert à rien, mais il semble que vous pouvez exploiter le code pour produire une carte qui permettent l'ouverture de la porte. Essez de trouvé un moyen le plus simple possible.

Le code provient d'éclipse en JavaCard. Vous disposez du projet complet dans ce répertoire `door.tar.xr`. Le code est aussi disponible ci-après mais vous pouvais surment passer à côté de certaine information ;).

```
package door;

import javacard.framework.*;

import javacard.framework.APDU;
import javacard.framework.Applet;
import javacard.framework.ISOException;
```

```
public class Door extends Applet {

    private final static byte ID_CARD_CLA =(byte)0x80;

    private static final byte VALIDATE_PIN_INS = 0x22;
    private static final byte GET_ID_INS       = 0x23;

    final static byte PIN_TRY_LIMIT = (byte) 0x03;
    final static byte MAX_PIN_SIZE = (byte) 0x08;
    private final static byte PIN_SIZE =(byte)0x04;

    private final static short SW_VERIFICATION_FAILED = 0x6300;
    private final static short SW_PIN_VERIFICATION_REQUIRED = 0x6301;

    OwnerPIN pin;
    byte[] id = {0x24}; // id de la carte qui serat donné au reader

    private Door() {
        // initialisation du code pin de la carte
        pin = new OwnerPIN(PIN_TRY_LIMIT, MAX_PIN_SIZE);
        pin.update(new byte[]{0x01,0x05,0x08,0x03},(short) 0, PIN_SIZE);
        register();
    }

    public static void install(
        byte bArray[],
        short bOffset,
        byte bLength) throws ISOException {
        new Door().register();
    }

    public void process(APDU apdu) throws ISOException {
        byte[] buffer = apdu.getBuffer();

        if(this.selectingApplet()) return;

        if (buffer[ISO7816.OFFSET_CLA] != ID_CARD_CLA) // quitter si mauvais CLA
            ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

        switch(buffer[ISO7816.OFFSET_INS]){
            case VALIDATE_PIN_INS:
                validatePin(apdu); break;
            case GET_ID_INS:
                getId(apdu); break;
            default: ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
```

```
    }

    public void validatePin(APDU apdu) {
        byte[] buffer = apdu.getBuffer();
        if(buffer[ISO7816.OFFSET_LC]==PIN_SIZE){
            apdu.setIncomingAndReceive();
            if(pin.check(buffer, ISO7816.OFFSET_CDATA,PIN_SIZE)==false)
                ISOException.throwIt(SW_VERIFICATION_FAILED);
        }else ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    }

    public void getId(APDU apdu) {
        if(!pin.isValidated()) ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);

        apdu.setOutgoing();
        apdu.setOutgoingLength((short) 1);
        apdu.sendBytesLong((byte[]) id, (short) 0, (short) 1);
    }

}
```

Petit message