

SAE - La Carotte Électronique

Note

Ce projet est à réaliser les de cadres des SAÉs suivantes :

- SAÉ 5.01 - Concevoir, réaliser et présenter une solution technique
- SAÉ 5.02 - Piloter un projet informatique

Compétences ciblées

- Créer des outils et applications informatiques pour les R&T
- Utilisation du système de gestion de versions GIT et la plateforme GitHub ou GitLab pour une application porte-monnaie électronique sur cartes à puce

Livrables attendus

- Étudier standard ISO 7816 et réaliser une application réseau pour piloter un porte-monnaie électronique sur carte à puce tout en respectant les normes de cybersécurité.
- Étudier les différentes vulnérabilités des applications et apporter une solution adéquate pour sécuriser l'accès
- Utiliser correctement le système de gestion de versions et les plateformes de gestion de versions
- Travailler en collaboration avec les membres de l'équipe



Sommaire :

Introduction	3
Git	4
Deuxième étape : La planification	5
Première partie : Programmer la carte à puce (Rubrovitamin) :	6
MAJ 15/11/2023 Nouvelle fonction : Insérer le numéro étudiant	8
MAJ 22/11/2023 Mise à jour du nombre d'octets: 2 à 4 octets	8
Deuxième partie Personnalisation (Lubiana) :	9
Première fonction : Affichage de la version de la carte	9
Deuxième fonction : Attribution de la carte	10
Troisième fonction : Affichage des données de la carte	10
Quatrième Fonction : Affichage du solde de la carte	11
Cinquième Fonction : Mettre le solde initial	12
Sixième Fonction : Réinitialiser le solde initial	13
MAJ 15/11/2023 Septième Fonction : Insérer le numéro étudiant	13
MAJ 20/11/2023 Huitième Fonction : Réinitialiser le solde à chaque activation du solde initial	14
MAJ 23/11/2023 : Mise à jour du nombre d'octets: 2 à 4 octets	14
Troisième partie : base de données	15
Quatrième et Cinquième partie : Rodelika	17
1.Rodelika version base	17
2.Rodelika version web	17
Sixième partie : Borne de recharge	18
Première fonction : Afficher les informations personnelles	18
Deuxième fonction : Consulter les bonus	18
Troisième fonction : Transférer les bonus	19
Quatrième fonction : Consulter le crédit sur la carte :	19
Cinquième fonction : Recharger le crédit avec une carte bancaire:	20
MAJ 23/11/2023 : Mise à jour du nombre d'octets: 2 à 4 octets	21
Réseaux Inter-VM :	21
Septième partie - Machine à café (Cafedelika) :	21
Première version - Machine à café basique :	21
Deuxième version - Machine à café amélioré :	22
Troisième version - Machine à café connecté :	22
Quatrième version - Machine à café finale et sécurisée:	24
MAJ 23/11/2023 : Mise à jour du nombre d'octets: 2 à 4 octets	26
Huitième partie - Vulnérabilités :	26
Rubrovitamin.c :	26
Lubiana.py :	27
Cafedelika.py :	28
Fonctionnement Général	29
Relations avec d'Autres Applications / Modes de Communication	29

Vulnérabilités, Dysfonctionnements ou Attaques Possibles	30
Solutions Proposées	30
Berlicum.py :	30
Fonctionnement Général	30
Relations avec d'Autres Applications / Modes de Communication	31
Vulnérabilités, Dysfonctionnements ou Attaques Possibles	32
Solutions Proposées	32
Méthode de sécurité pour la carte :	33
Conclusion :	33

Introduction

L'IUT de Vélizy se lance dans un projet novateur, baptisé "La Carotte Électronique", visant à récompenser les étudiants méritants par le biais d'un système de récompenses financières. Ce concept innovant permet aux étudiants de convertir leurs mérites en crédits utilisables dans des distributeurs de boissons chaudes disséminés sur le campus.

L'initiative comprend la mise en place de distributeurs de boissons chaudes équipés de lecteurs de cartes à puce, ainsi qu'un bureau administratif dédié à la gestion du projet. Lors de la personnalisation des cartes, les étudiants voient leur identité, numéro d'étudiant, nom, prénom, et le solde initial de leur carte intégrés. De plus, des bonus peuvent être attribués par les enseignants pour encourager la participation et l'excellence académique.

Le projet s'étend également à une borne de recharge centralisée, où les étudiants peuvent consulter leur solde, transférer des bonus, recharger leur carte, et initier le transfert du crédit initial.

Pour concrétiser ce système, nous sommes appelé à développer les logiciels centraux, dont Rubrovitamina (intégré dans la carte), Lubiana (pour la personnalisation), Rodelika (gestion des récompenses), Berlicum (la borne de recharge), Purple Dragon (la base de données) et Cafédélika (machine à café).

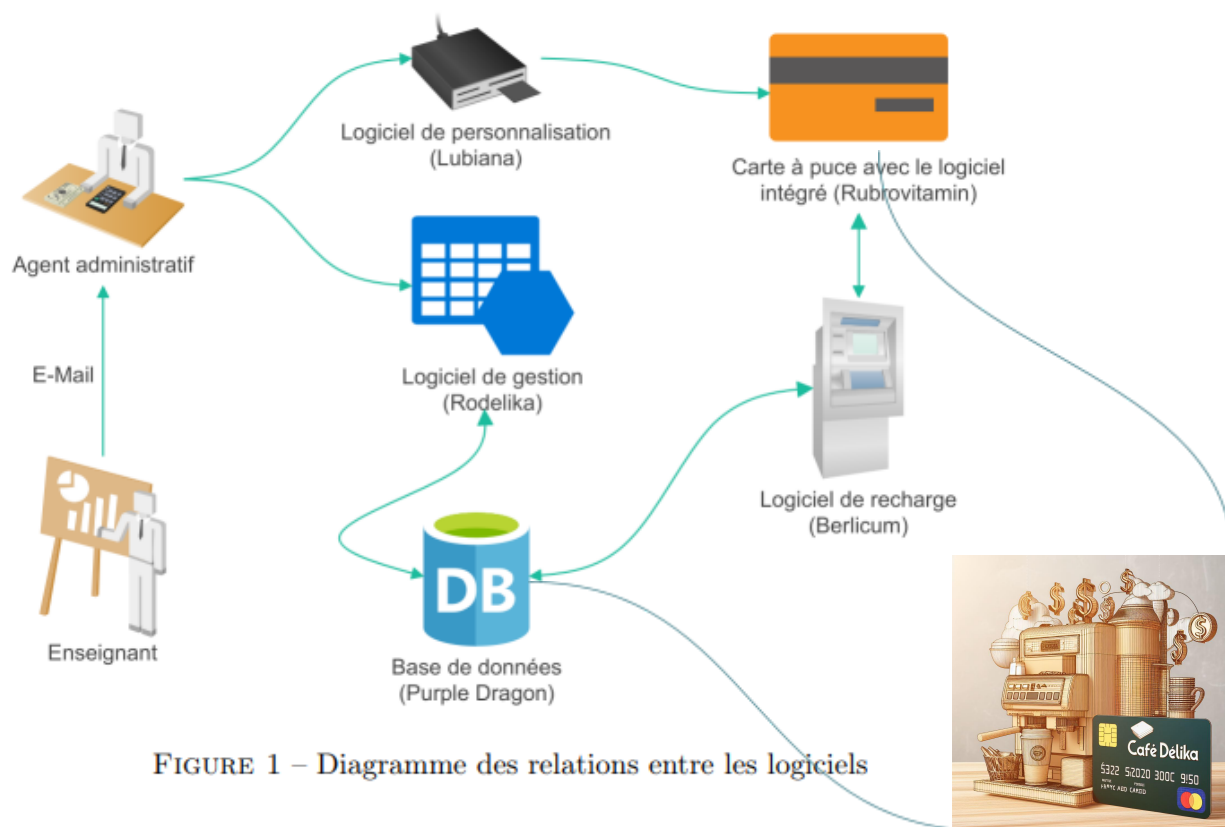


FIGURE 1 – Diagramme des relations entre les logiciels

Notre rôle dans ce projet est de concevoir ces éléments logiciels, avec un accent particulier sur la sécurité et l'efficacité. Cette collaboration requiert une organisation en équipe, l'utilisation de GIT pour la gestion de versions, et une approche méthodique pour atteindre les objectifs fixés.

Le rapport détaillé ci-dessous décompose chaque composant du projet et fournit des indications sur la marche à suivre.

Que le projet La Carotte Électronique commence !

Gestion de Version avec Git dans le Projet



La gestion de version et la collaboration est une pratique essentielle dans le développement logiciel moderne, permettant de suivre les modifications apportées au code source tout au long du projet. Dans le cadre du projet "La Carotte Électronique" à l'IUT de Vélizy, nous avons utilisé Git comme système de gestion de versions distribuées.

Notre Github : <https://github.com/ceZarMax/I.U.T-3rd-Year-SAE-Carte-a-puce>

Qu'est-ce que Git ?

Git est un système de gestion de versions décentralisé, créé par Linus Torvalds. Il permet de suivre les changements dans le code source, de collaborer efficacement avec d'autres développeurs, et de gérer les différentes versions du projet.

Utilisation de Git dans le Projet :

L'utilisation de Git dans le projet a permis une collaboration fluide entre les membres de l'équipe. Chaque aspect du développement, des ajustements de code aux nouvelles fonctionnalités, a été suivi et commenté grâce à Git.

Avantages de l'utilisation de Git :

- Historique des Modifications : Git a enregistré chaque modification apportée au code, facilitant le suivi des évolutions du projet.
- Collaboration Efficace : Les membres de l'équipe ont pu travailler simultanément sur différentes parties du projet sans conflits majeurs, grâce aux fonctionnalités de branches et fusion de Git.
- Sauvegardes Régulières : Les "commits" réguliers ont assuré des sauvegardes fréquentes du code, offrant une sécurité contre la perte de données.

Plateforme d'Hébergement :

Le dépôt Git du projet a été hébergé sur une plateforme en ligne, offrant un accès centralisé au code source.

En conclusion, l'utilisation de Git dans le projet "La Carotte Électronique" a grandement contribué à la réussite du développement logiciel en assurant une collaboration harmonieuse, un suivi précis des modifications, et une gestion efficace des versions du code source.

La planification

Rapport

Ajouter photo du code pour la PARTIE 1

14 nov. 4/4 MS

Ajouter résultat partie 2

14 nov. 4/4 MS

Partie de la machine à café

5/5

Faire le powerpoint

7/7 MS

Rapport

8/8 MS

+ Ajouter une carte

Première Partie : Programmer la carte à puce

Terminer le code sur la carte à puce

1/1 MS

créer la classe d'instruction 0x81 :

2/2 MS

créer la classe d'instruction 0x82 :

2/2 MS

Check EEPROM read bytes des fonctions soldes

1/1 MS

Créer la fonction en C : Reinit_Solde

1/1 MS

V2 : Créer une fonction pour implémenter un ID

1/1 MS

Mise à jour du solde de 2 à 4 octets

1/1 G

+ Ajouter une carte

Deuxième partie : Personnalisation

Terminer le logiciel Lubiana pour l'administrateur

10 nov. 5/5 MS

créer un menu interactif qui interagit via le clavier

1/1 MS

Créer la fonction Reinitialiser Solde

1/1 MS

V2 : Créer une fonction pour que l'admin ajoute l'identifiant de la carte lors de l'attribution

1/1 MS

Mise à jour : 2 à 4 octets par rapport à la maj de la carte

1/1 G

+ Ajouter une carte

Troisième partie : Base de données

Implémenter la base de données

4/4 G QH SB MS

+ Ajouter une carte

Quatrième partie : logiciel de gestion Rodelika

créer un menu interactif

QH

Les différentes fonctionnalités à implémenter :

QH

+ Ajouter une carte

Cinquième partie : Application de gestion sur web

Prendre une template js sur internet

QH

Relier la partie SQL au site et logiciel python

0/2 QH

+ Ajouter une carte

sixième partie : Borne de recharge

implémenter ces fonctionnalités sur la borne :

6/6 G QH

+ Ajouter une carte

Septième partie : Machine à café

Créer la partie débit sur le programme C

1/1 MS

Créer le logiciel de simulation de la machine à café en python

4/4 MS

Mise à jour : 2 à 4 octets par rapport à la maj de la carte

1/1 G

+ Ajouter une carte

Huitième partie : Vulnérabilités

— Anti-arrachement (obligatoire pour ce projet) : si un arrachement de carte est survenu lors d'une transaction, il faut remettre la carte dans un état correcte lors de la prochaine utilisation de la carte.

— Anti-rejoue (facultatif pour ce projet) : pour éviter les ajouts par les étudiants, on a fait en sorte que le solde de la BDD et celui de la carte correspondent, sinon ils ne peuvent l'utiliser.

0/1 MS

Explications de chaque logiciel

6/6

+ Ajouter une carte

Première partie : Programmer la carte à puce (Rubrovitamin) :

Nous allons devoir programmer la carte à puce pour permettre d'attribuer la carte à un étudiant, c'est-à-dire écrire d'une façon persistante les informations de l'étudiant, par exemple son nom, son prénom et son numéro d'étudiant. Le processus de personnalisation est géré par l'agent administratif via un lecteur de carte à puce et le logiciel lubiana codé en python que nous allons développer dans la prochaine partie.

Tout d'abord, nous allons devoir créer deux classes dans notre script en c.

CLS	Description	INS	Description	Type	P1	P2	Lc	Data	Le
0x81	Classe de personnalisation	0x00	Version de l'application	Sortie	0x00	0x00			0x04
		0x01	Entrer personnalisation	Entrée	0x00	0x00	0x07	0x43 0x61 0x72 0x72 0x6f 0x74 0x65	
		0x02	Lire personnalisation	Sortie	0x00	0x00			0x07
0x82	Classe de gestion de paiement	0x01	Lire le solde de la carte	Sortie	0x00	0x00			0x02
		0x02	Ajouter solde 1.00 €	Entrée	0x00	0x00	0x02	0x00 0x64	
		0x03	Dépense 0.20 €	Entrée	0x00	0x00	0x02	0x00 0x14	

Pour les opérations de personnalisation, nous allons créer la classe d'instructions **0x81** avec trois instructions :

- L'instruction 0x00 permet de retourner la version de la carte.
Exemple : 81 00 00 00 04
- L'instruction 0x01 permet d'entrer les informations de l'étudiant.
- L'instruction 0x02 permet de lire les informations de l'étudiant.

Une autre classe d'instruction **0x82** qui permet de gérer les transactions :

- L'instruction 0x01 permet de retourner le solde de la carte.
- L'instruction 0x02 permet de créditer la carte.
- L'instruction 0x03 permet de débiter la carte.

Cette première partie du rapport se concentre sur la programmation de la carte à puce et la description des fonctionnalités de base du programme. Le code source du programme est écrit en langage C pour les microcontrôleurs AVR, et il utilise les bibliothèques standard pour les opérations de lecture/écriture dans l'EEPROM, la gestion des données, ainsi que les fonctions d'entrée/sortie.

Le programme "Rubrovitamin" est structuré en plusieurs parties, chacune ayant un rôle spécifique :

1. Définition de la version : Le programme commence par définir une version du logiciel. Cette version est stockée dans la mémoire **FLASH** de la carte à puce et peut être lue par le système d'accès contrôlé ou le logiciel python qu'on a développé : Lubiana.
2. Gestion de la personnalisation : La carte à puce peut être personnalisée avec des données spécifiques. Six fonctions, `intro_nom` et `lire_nom`, prenom et date de naissance, permettent d'introduire des données dans l'**EEPROM** de la carte et de les lire ultérieurement. Cela permet de personnaliser la carte selon les besoins du système d'accès contrôlé.

```
void intro_nom(){ // Fonction de personnalisation, données écrites dans l'EEPROM
    int i;
    unsigned char data_nom[MAX_PERSO];
    // vérification de la taille
    if (p3>MAX_PERSO){
        sw1=0x6c; // P3 incorrect
        sw2=MAX_PERSO; // sw2 contient l'information de la taille correcte
        return;
    }
    sendbytet0(1); // acquitement
    for(i=0;i<p3;i++){ // boucle d'envoi du message
        data_nom[i]=recbytet0();
    }
    eeprom_write_block(data_nom,ee_nom,p3);
    eeprom_write_byte(&ee_taille_nom,p3);
    sw1=0x90;
}
```

```
void lire_nom(){
    int i;
    char buffer[MAX_PERSO];
    uint8_t taille;
    taille=eeprom_read_byte(&ee_taille_nom);
    if (p3!=taille){
        sw1=0x6c; // P3 incorrect
        sw2=taille;
        return;
    }
    sendbytet0(1);
    eeprom_read_block(buffer, ee_nom, taille);

    for (i=0;i<p3;i++){
        sendbytet0(buffer[i]);
    }
    sw1=0x90;
}
```

3. Gestion des paiements : La carte "Rubrovitamin" permet de gérer un solde et d'effectuer des opérations de crédit et de débit. Les fonctions `LectureSolde`, `credit` et `Depenser` gèrent ces opérations. Les données relatives au solde sont stockées en mémoire **EEPROM**.
4. ATR (Answer To Reset) : La procédure ATR est utilisée pour initialiser la communication entre la carte et le système d'accès contrôlé. L'ATR de la carte est généré et envoyé au système.

5. Structure du programme : Le programme principal est basé sur une boucle infinie qui lit les commandes entrantes, les traite en fonction de la classe et de l'instruction, et renvoie un statut de sortie approprié. Les classes `0x81` et `0x82` sont utilisées pour la personnalisation et la gestion des paiements, respectivement.

6. Ajout de la fonction de réinitialisation du solde: De base, il n'y a pas de fonction de réinitialisation de solde. Je vais le faire pour que l'administrateur puisse réinitialiser le solde au cas où il en aurait besoin :

```
void Reinit() {  
    if(p3 != 2){  
        sw1 = 0x6c ;  
        sw2 = 2;  
        return ;  
    }  
    uint16_t solde_initial = 0; // Montant initial du solde (0 euros)  
    eeprom_write_word(&solde, solde_initial);  
    sw1 = 0x90; // Succès  
}
```

Dans cette première partie, nous avons examiné les fonctionnalités de base du programme "Rubrovitamin" et la manière dont il communique avec le système d'accès contrôlé. Les prochaines sections du rapport exploreront en détail les différentes parties du programme et fourniront des exemples d'utilisation dans le logiciel Administrateur.

MAJ 15/11/2023 Nouvelle fonction : Insérer le numéro étudiant

Au vue d'un problème lié à la machine, nous allons faire en sorte que l'administrateur puisse insérer un numéro étudiant. Ceci permettra de faciliter la tâche de la version finale de la machine à café, qui sera liée à notre BDD et consistera à envoyer les requêtes SQL lors d'une vente de boissons etc.

MAJ 22/11/2023 Mise à jour du nombre d'octets: 2 à 4 octets

Après quelques problèmes rencontrés sur Berlicum, la borne de recharge, nous avons compris que nous avons eu besoin de plus de stockage sur la carte afin de pouvoir être sûr que les transactions puissent bien fonctionner.

Deuxième partie Personnalisation (Lubiana) :

Le processus de personnalisation comporte deux étapes :

- Mettre les informations personnelles de l'étudiant sur la nouvelle carte
- Attribuer le solde initial de 1.00 €

On va commencer à créer étape par étape le logiciel python permettant de :

- 1 - Afficher la version de carte
- 2 - Afficher les données de la carte
- 3 - Attribuer la carte
- 4 - Mettre le solde initial
- 5 - Consulter le solde
- 6 - Quitter

Tout d'abord, je vais importer les librairies smartcard python pour la gestion des cartes à puce et la communication avec les lecteurs .

Ensuite, on va créer et modifier la fonction *init_smart_card*. Permettant de détecter en premier temps si un lecteur de carte est branché et en deuxième temps s'il y'a une carte dedans. Ensuite j'ai décidé de stocker l'atr dans une fonction globale *card_atr* pour print l'atr de manière plus lisible dans le main.

Si c'est tout bon et que la fonction *init_smart_card* a tout détecter, alors on continue avec le message de bienvenue. Qui se suivra de l'affichage du menu.

Et pour finir dans le **core** de notre code, nous avons la partie main V1, qui exécutera toutes nos fonctions qui fait de ce code un programme structuré. Nous ajouterons nos nouveaux choix et fonctions au fur et à mesure du temps.

Première fonction : Affichage de la version de la carte

L'affichage de la version de la carte constitue la première fonctionnalité de Rubrovitamin. Cette étape, intégrée au cœur du programme, vise à transmettre l'instruction nécessaire pour récupérer la version actuelle de la carte à puce. La version, initialement définie à 1.00 dans le programme C, est récupérée à partir de la mémoire flash de la carte. Cette fonction est fondamentale pour assurer la cohérence des mises à jour et des évolutions du logiciel embarqué, permettant ainsi un suivi précis de l'évolution du système. Dans ce processus, Rubrovitamina établit une première interaction significative avec la carte à puce, jetant les bases de notre système

.

Partie Résultat :

```
ATR (HEX) : 3B FC 01 05 05 00 00 48 65 6C 6C 6F 20 73 63 61 72 64
ATR (ASCII): ;.....Hello scard

1 - Afficher la version de carte
2 - Afficher les données de la carte
3 - Attribuer la carte
4 - Mettre le solde initial
5 - Consulter le solde
6 - Quitter
Choix : 1

sw1 : 0x90 |
sw2 : 0x00 |
Version de la carte : 1.00
```

Deuxième fonction : Attribution de la carte

L'attribution de la carte représente une étape essentielle du processus de personnalisation de Rubrovitamina. Cette fonctionnalité clé permet d'envoyer les instructions nécessaires à la carte à puce pour écrire dans l'EEPROM, garantissant ainsi la conservation des données cruciales. Lorsqu'un étudiant se présente pour recevoir sa carte personnalisée, le programme vérifie d'abord la taille des données introduites, assurant ainsi la cohérence des informations. Une fois cette vérification effectuée, Rubrovitamina transmet les données, dont le numéro d'étudiant, le nom, le prénom, et initialise la carte à 0.00 euros. Cette procédure, intégrée de manière transparente dans le flux global du programme, facilite la distribution des cartes tout en établissant les bases nécessaires à une utilisation efficace du système de récompense La Carotte Électronique.

Partie Résultat :

```
Choix : 3
Saisissez le Nom de l'élève : Arvin-Berod
Affichage de l'APDU : [129, 1, 0, 0, 11, 65, 114, 118, 105, 110, 45, 66, 101, 114, 111, 100]

sw1 : 0x90 | sw2 : 0x00
Succès !
Nom de l'élève : Arvin-Berod
Saisissez le Prénom de l'élève : Maxence
Affichage de l'APDU : [129, 3, 0, 0, 7, 77, 97, 120, 101, 110, 99, 101]

sw1 : 0x90 | sw2 : 0x00
Succès !
Prénom de l'élève : Maxence
Saisissez la date de naissance (Format : JJMMAAAA) : 15122003
Affichage de l'APDU : [129, 5, 0, 0, 8, 49, 53, 49, 50, 50, 48, 48, 51]

sw1 : 0x90 | sw2 : 0x00
Succès !
Date de naissance : 15/12/2003

---

1 - Afficher la version de carte
2 - Afficher les données de la carte
3 - Attribuer la carte
4 - Mettre le solde initial
5 - Consulter le solde
6 - Quitter
Choix : █
```

J'ai rajouté le solde de la carte dans l'affichage, pour bien voir toutes les données ensembles.

```
Choix : 2

sw1 : 0x6C |
sw2 : 0x0B |

sw1 : 0x90 |
sw2 : 0x00 |
Nom : Arvin-Berod

sw1 : 0x6C |
sw2 : 0x07 |

sw1 : 0x90 |
sw2 : 0x00 |
Prénom : Maxence

sw1 : 0x6C |
sw2 : 0x08 |

sw1 : 0x90 |
sw2 : 0x00 |
Date de naissance : 15/12/2003
sw1 : 0x90 | sw2 : 0x00

sw1 : 0x90 |
sw2 : 0x00 |
Solde de la carte : 1.00 €
```

Quatrième Fonction : Affichage du solde de la carte

Voici la fonction permettant d'afficher le solde (je la réutilise lors de l'affichage des données de la carte).

Nous envoyons l'instruction à la carte qui nous renvoie les données du solde converties en centimes pour l'afficher correctement.

Partie Résultat :

```
---
1 - Afficher la version de carte
2 - Afficher les données de la carte
3 - Attribuer la carte
4 - Mettre le solde initial
5 - Consulter le solde
6 - Quitter
Choix : 5
sw1 : 0x90 | sw2 : 0x00

sw1 : 0x90 |
sw2 : 0x00 |
Solde de la carte : 1.00 €

---
```

Cinquième Fonction : Mettre le solde initial

Lorsqu'un étudiant récupère sa carte pour la première fois, cette fonctionnalité lui attribue automatiquement un crédit initial de 1.00 euro, environ cinq boissons. Cette incitation vise à encourager les étudiants à adopter le système de récompense dès le départ. En déployant cette logique incitative au sein de Rubrovitamina, le programme assure une expérience utilisateur fluide et positive, encourageant ainsi l'adoption du système et récompensant les étudiants dès leur adhésion initiale.

Partie Résultat :

```
---  
1 - Afficher la version de carte  
2 - Afficher les données de la carte  
3 - Attribuer la carte  
4 - Mettre le solde initial  
5 - Consulter le solde  
6 - Quitter  
Choix : 4  
  
sw1 : 0x90 | sw2 : 0x00  
Succès !  
Crédit initial ajouté : 1.0 €
```

```
---  
1 - Afficher la version de carte  
2 - Afficher les données de la carte  
3 - Attribuer la carte  
4 - Mettre le solde initial  
5 - Consulter le solde  
6 - Quitter  
Choix : 5  
sw1 : 0x90 | sw2 : 0x00  
  
sw1 : 0x90 |  
sw2 : 0x00 |  
Solde de la carte : 2.00 €  
---
```

Si j'affiche le solde, on voit qu'on a 2€. Le fait qu'il y est déjà un montant crédité sur la carte, il va s'ajouter, mais si l'administrateur veut réinitialiser le solde à 0 comment il fait ?

C'est le problème que je vais régler en créant une 6ème option permettant de faire ceci.

Sixième Fonction : Réinitialiser le solde initial

En cas de nécessité, notamment lorsque l'administrateur souhaite remettre à zéro le solde initial d'une carte, cette option offre une solution efficace. En utilisant cette fonction, l'administrateur peut réinitialiser le solde initial à 0.00 euro, redonnant ainsi à la carte son statut initial.

Partie Résultat :

On a 2€ dans la carte :

```
7 - Quitter
Choix : 5
sw1 : 0x90 | sw2 : 0x00

    sw1 : 0x90 |
    sw2 : 0x00 |
    Solde de la carte : 2.00 €

---
```

On va réinitialiser le solde :

```
1 - Afficher la version de carte
2 - Afficher les données de la carte
3 - Attribuer la carte
4 - Mettre le solde initial
5 - Consulter le solde
6 - Réinitialiser le solde
7 - Quitter
Choix : 6

sw1 : 0x90 | sw2 : 0x00
Succès !
Le crédit a été réinitialisé à 0 €

---
```

MAJ 15/11/2023 Septième Fonction : Insérer le numéro étudiant

Au vue d'un problème lié à la machine, nous allons faire en sorte que l'administrateur puisse insérer un numéro étudiant.

Lorsqu'un étudiant se présente pour la première fois afin de récupérer sa carte, l'agent administratif peut saisir et enregistrer le numéro étudiant dans la puce électronique. Cette information, essentielle pour identifier de manière précise chaque utilisateur, est ensuite stockée de manière sécurisée dans la mémoire EEPROM de la carte à puce. Ainsi, l'agent devra aussi (à l'aide de rodelika) ajouter le même numéro étudiant sur rodelika.

Partie Résultat :

```
---  
1 - Afficher la version de carte  
2 - Afficher les données de la carte  
3 - Attribuer la carte  
4 - Mettre le solde initial  
5 - Consulter le solde  
6 - Réinitialisé le solde  
7 - Quitter  
Choix : 3  
Saisissez le numéro étudiant (MAX : 999) : 11  
Affichage de l'APDU : [129, 7, 0, 0, 2, 49, 49]
```

MAJ 20/11/2023 Huitième Fonction : Réinitialiser le solde à chaque activation du solde initial

En cas de besoin de réinitialisation, notamment lorsque l'administrateur souhaite remettre à zéro le solde initial pour un étudiant, cette fonctionnalité offre une solution simple et efficace. Lorsque l'administrateur effectue un ajout de solde initial, le système déclenche automatiquement la réinitialisation du solde, ramenant ainsi le crédit initial à sa valeur par défaut. Cette approche garantit une uniformité dans le processus de distribution des récompenses tout en permettant à l'IUT de maintenir un contrôle précis sur les montants attribués aux étudiants, assurant ainsi une gestion transparente et équitable du programme de récompenses.

MAJ 23/11/2023 : Mise à jour du nombre d'octets: 2 à 4 octets

Suite au passage de 2 à 4 octets sur notre carte à puce, il fallait apporter quelques correctifs afin de garantir le bon fonctionnement de la lecture de la carte, notamment sur le solde qui allait être mis sur 4 octets.

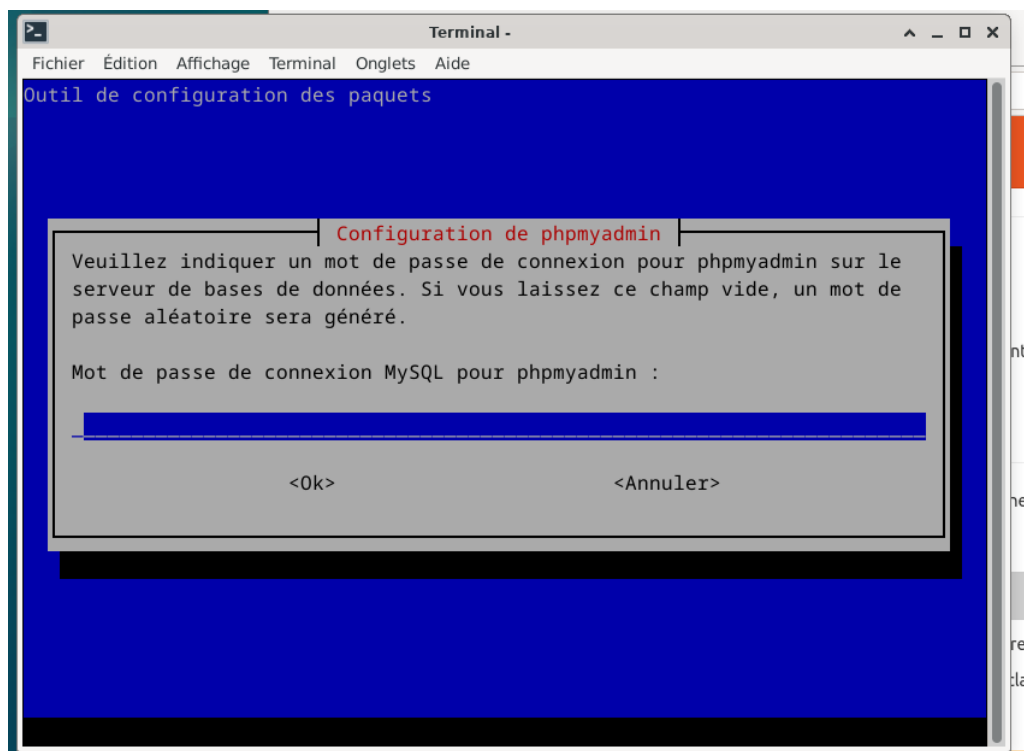
Troisième partie : base de données

Pour cette partie on utilise Mysql et phpadmin afin de créer des requêtes, et les visualiser.

on utilisera le mot de passe root afin de pouvoir se connecter à la base mysql :

afin de pouvoir installer phpmyadmin on suit les commandes suivantes :

```
sudo apt install phpmyadmin
```



```
sudo systemctl restart apache2
```

php étant, pas installé par défaut sur la machine il faut le télécharger :

```
sudo apt install php
```

La configuration de php est nécessaire afin de garantir le bon fonctionnement de php et phpmyadmin.

```
sudo nano /etc/apache2/mods-available/php7.4.conf
```

Il faut ajouter ces deux lignes dans le fichier :

```
AddHandler php7.4 .php
```

```
Include /etc/phpmyadmin/apache.conf
```

→ ici, c'est la ligne à ajouter dans "php7.4.conf".

```
sudo systemctl restart apache2
```

une fois toutes les étapes réalisées, on peut lancer phpmyadmin via l'adresse suivante :

<http://localhost/phpmyadmin/index.php>

Quatrième et Cinquième partie : Rodelika

1.Rodelika version base

Rodelika est un programme python qui permet de gérer la base de données liées aux utilisateurs des cartes. Ce programme a pour but d'être utilisé par l'administrateur des cartes qui va pouvoir ainsi créer des nouveaux utilisateurs de la carte, consulter des informations de la base de données et ajouter des bonus.

Les fonctionnalités de ce programme (relié à la base de données Purple Dragon vu précédemment) sont de:

- Afficher la liste de l'ensemble des étudiants (qui ont une carte)
- Afficher le solde total de chaque étudiant
- Ajouter un nouvel étudiant tout en initialisant son compte avec un bonus de +1.00€
- Attribuer un bonus de +1.00€ à un étudiant.

2.Rodelika version web

Afin de faciliter l'accessibilité à l'application Rodelika pour les utilisateurs finaux, on propose une version web de notre application (qui sera accessible en ligne dans le cadre d'un produit fini, installée sur un serveur).

Elle permet à l'administrateur de la carte de gérer de manière plus claire et visuelle l'ensemble des fonctionnalités disponibles sur la version précédente. De plus, un enseignant pourra directement ajouter un bonus à un élève sans passer par une demande à l'administrateur.

L'application reprend les fonctionnalités précédentes en utilisant la librairie Flask pour afficher des pages web qui utilisent des fonctions python.



Flask est un micro framework open-source de développement web en Python. Il est extrêmement rapide et simple à mettre en place et se base sur des principes très légers

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def FONCTION():
    return "Hello world !"
if __name__ == "__main__":
    app.run()
```

On doit simplement ajouter quelque chose de cette forme de notre code et définir des routes qui appelleront des fonctions qu'on aura défini.

On propose donc d'avoir:

- une page d'accueil (ipduseur:5000/)
- un menu principal (ipduseur:5000/main_menu)
- différentes pages qui feront toutes les fonctions de Rodelika
(ipduseur:5000/list_student)
(ipduseur:5000/list_student_with_sold)
(ipduseur:5000/add_student)
(ipduseur:5000/add_bonus)

Notre application est donc bien évidemment toujours reliée à notre même BDD.

Sixième partie : Borne de recharge

On va créer un logiciel qui va simuler une borne de recharge qui va permettre aux étudiants possédant une carte afin de pouvoir faire les actions suivantes :

Recharger le crédit avec carte bancaire :

Première fonction : Afficher les informations personnelles

Cette opération permet d'afficher les informations personnelles inscrites sur la carte à puce.

Les informations seront lues dans les données de la carte à puces

```
-----MES INFORMATIONS-----  
  
Nom : Girault  
  
Prénom : Adrien  
  
Date de naissance : 30/06/2003  
-----
```

Deuxième fonction : Consulter les bonus

Permet de consulter les bonus attribués mais pas encore transférés sur la carte.
Une requête SQL sera effectuée sur la base de données pour cela.

```
Voici le montant de vos bonus : 0.00 €
```

Troisième fonction : Transférer les bonus

Permet de transférer les bonus disponibles sur la carte et inscrire dans la base de données que les bonus sont bien transférés.

→ deux choses sont à faire ici :

- Il faut ajouter le bonus sur la carte dans un premier temps donc
Nouveau solde = Ancien Solde + Bonus Disponible






(ici on ajoute un bonus pour vérifier que cela fonctionne) :

```
Choix : 3
Total des bonus disponibles : 1.0 €
Entrez le montant du bonus à transférer : 1.0
Bonus de 1.0 € transférés avec succès.
```

Voilà à présent, la base de donnée :

							▼ etu_num	etu_nom	etu_prenom	etu_solde ▲ 1
<input type="checkbox"/>		Éditer		Copier		Supprimer	1	Arvin-Berod	Maxence	0
<input type="checkbox"/>		Éditer		Copier		Supprimer	2	Hubert	Quentin	0
<input type="checkbox"/>		Éditer		Copier		Supprimer	3	El Beki	Sohayb	0
<input type="checkbox"/>		Éditer		Copier		Supprimer	4	Girault	Adrien	1

on remarque ici, que l'étudiant 4 a bien été crédité de 1 euro de bonus :

<input type="checkbox"/>		Éditer		Copier		Supprimer	2	2023-05-16 08:23:32	1.00	Initial	Bonus
<input type="checkbox"/>		Éditer		Copier		Supprimer	3	2023-05-18 15:26:38	1.00	initial	Bonus
<input type="checkbox"/>		Éditer		Copier		Supprimer	2	2023-07-04 15:24:35	1.00	bonne initiative	Bonus
<input type="checkbox"/>		Éditer		Copier		Supprimer	4	2023-07-13 15:27:04	0.00	intial	Bonus
<input type="checkbox"/>		Éditer		Copier		Supprimer	1	2023-11-16 16:20:43	0.40	Cappuccino	Dépense
<input type="checkbox"/>		Éditer		Copier		Supprimer	4	2023-11-28 00:35:20	-1.00	Recharge par borne	Bonus transféré

et que le bonus a bien été débité pour ne pas être avoir des bonus "infini".

Quatrième fonction : Consulter le crédit sur la carte :

Permet simplement à un étudiant de consulter son solde disponible sur sa carte.
Cette information est lu sur la carte à puce:

```
Choix : 4
-----MON SOLDE -----
Solde de la carte : 1.00 €
-----
```

Cinquième fonction : Recharger le crédit avec une carte bancaire:

Il se peut qu'un étudiant ait épuisé ses bonus, cette opération permet de recharger le crédit de la carte à l'aide de la carte bancaire. Pour ce projet le processus de recharge avec une carte bancaire est factice, il faut simplement afficher des messages simulant la bonne transaction de la carte bancaire.

Pour cela il faut faire encore une fois deux choses :







Dans un premier temps, il faut recharger la carte, avec le même principe du transfert de bonus :

Nouveau solde = Ancien solde + Recharge Crédit bancaire

Dans un deuxième temps, il faut envoyer une requête SQL sur la base de données pour mettre à jour l'ancien solde.

```
Choix : 5
Entrez le montant à ajouter (en euros, max 255) : 5
Succès ! Crédit de 5 € ajouté.
```

on retrouve bien les 5 euros sur la base de donnée :

<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	4	2023-11-28 00:35:20	-1.00	Recharge par borne	Bonus transféré
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	4	2023-11-28 00:43:42	5.00	Rechargement par carte	Recharge

ainsi que sur le solde de l'étudiant :

<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	4	Girault	Adrien	6
--------------------------	--	--	---	---	---------	--------	---

MAJ 23/11/2023 : Mise à jour du nombre d'octets: 2 à 4 octets

Suite au passage de 2 à 4 octets sur notre carte à puce, il fallait apporter quelques correctifs afin de garantir le bon fonctionnement de la lecture de la carte, notamment sur le solde qui allait être mis sur 4 octets.

Réseaux Inter-VM :

Nous allons devoir grouper toutes nos applications ensemble dans le même réseau pour simuler la machine à café, le pc admin, et la BDD. Elles seront sur un réseau privé interne. Cela mettra plus de réalisme et de contexte lors de la démo.

Septième partie - Machine à café (Cafedelika) :

On va créer un logiciel qui simule la machine à café étant donné qu'on a pas de machine à café à disposition.

Première version : La machine à café (notre logiciel) se contentera de débiter le solde de la carte, et de donner la boisson.

Deuxième version : Modifier le prix en fonction de la boisson et en rajouter d'autres.

Troisième version : Améliorer la machine pour qu'elle soit relié à la BDD et envoie les transactions réalisés. Afficher aussi les statistiques du nombres de boissons réalisés.

Quatrième version (VERSION finale) : Qu'elle soit dynamique et vérifie si le solde de la carte correspond au solde que le logiciel de recharge (Berlicum) a envoyé à la BDD.

Première version - Machine à café basique :

PARTIE LOGICIEL PYTHON CAFEDELIKA :

J'ai laissé la fonction `init_smart_card` pour la bonne initialisation de la carte dans le lecteur. J'ai pu utilisé et me servir de la fonction `Dépenser` de ma carte pour débiter l'argent qu'il y'a dedans.

Tout d'abord, un aperçu de la première version, permettant de choisir une boisson et de la préparer/débiter s'il y'a assez sur la carte :

PARTIE RESULTAT CAFEDELIKA :

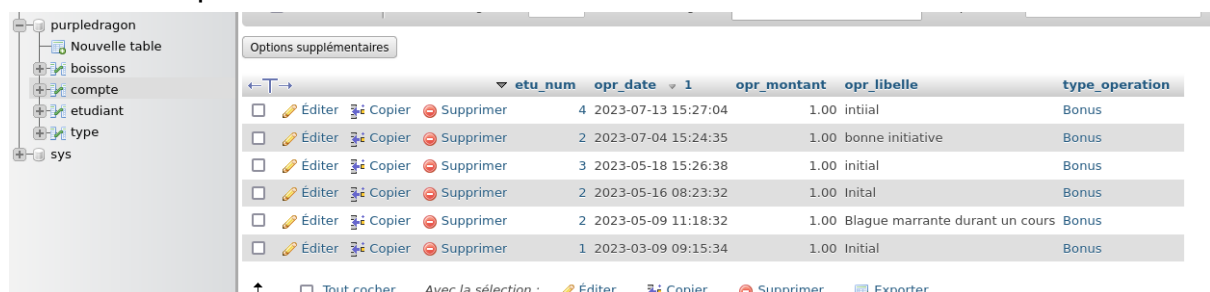
PARTIE BDD CAFEDELIKA :

Ma table boissons avant :



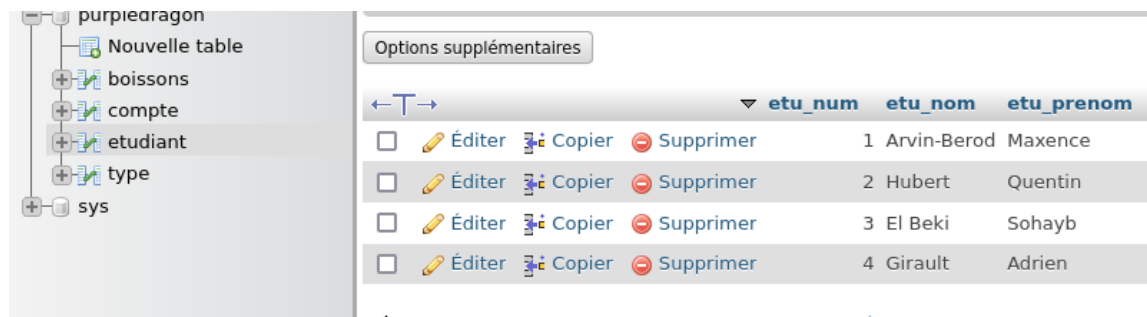
	boisson_id	boisson_nom	nombres_ventes	montant_total
<input type="checkbox"/>	1	Café	0	0
<input type="checkbox"/>	2	Café long	0	0
<input type="checkbox"/>	3	Cappuccino	0	0
<input type="checkbox"/>	4	Café BIO	0	0
<input type="checkbox"/>	5	Chocolat chaud	0	0
<input type="checkbox"/>	6	Thé	0	0

Ma table compte avant :



	etu_num	opr_date	opr_montant	opr_libelle	type_operation
<input type="checkbox"/>	4	2023-07-13 15:27:04	1.00	initial	Bonus
<input type="checkbox"/>	2	2023-07-04 15:24:35	1.00	bonne initiative	Bonus
<input type="checkbox"/>	3	2023-05-18 15:26:38	1.00	initial	Bonus
<input type="checkbox"/>	2	2023-05-16 08:23:32	1.00	Initial	Bonus
<input type="checkbox"/>	2	2023-05-09 11:18:32	1.00	Blague marrante durant un cours	Bonus
<input type="checkbox"/>	1	2023-03-09 09:15:34	1.00	Initial	Bonus

Ma table etudiant :



	etu_num	etu_nom	etu_prenom
<input type="checkbox"/>	1	Arvin-Berod	Maxence
<input type="checkbox"/>	2	Hubert	Quentin
<input type="checkbox"/>	3	El Beki	Sohayb
<input type="checkbox"/>	4	Girault	Adrien

PARTIE RESULTAT CAFEDELIKA :

Donc la, nous allons enregistrer la requête sql dans la BDD, mettre à jour les statistiques de ventes, et débiter sur la carte en dur le montant.

[illegible]

BDD Après :

purpledragon

- Nouvelle table
- boissons
- compte
- etudiant
- type
- sys

Options supplémentaires

	etu_num	opr_date	opr_montant	opr_libelle	type_operation
<input type="checkbox"/> Éditer Copier Supprimer	1	2023-11-16 16:20:43	0.40	Cappuccino	Dépense
<input type="checkbox"/> Éditer Copier Supprimer	4	2023-07-13 15:27:04	1.00	initial	Bonus
<input type="checkbox"/> Éditer Copier Supprimer	2	2023-07-04 15:24:35	1.00	bonne initiative	Bonus
<input type="checkbox"/> Éditer Copier Supprimer	3	2023-05-18 15:26:38	1.00	initial	Bonus
<input type="checkbox"/> Éditer Copier Supprimer	2	2023-05-16 08:23:32	1.00	Initial	Bonus
<input type="checkbox"/> Éditer Copier Supprimer	2	2023-05-09 11:18:32	1.00	Blague marrante durant un cours	Bonus
<input type="checkbox"/> Éditer Copier Supprimer	1	2023-03-09 09:15:34	1.00	Initial	Bonus

purpledragon

- Nouvelle table
- boissons
- compte
- etudiant
- type
- sys

Options supplémentaires

	boisson_id	boisson_nom	nombres_ventes	montant_total
<input type="checkbox"/> Éditer Copier Supprimer	1	Café	0	0
<input type="checkbox"/> Éditer Copier Supprimer	2	Café long	0	0
<input type="checkbox"/> Éditer Copier Supprimer	3	Cappuccino	1	0.4
<input type="checkbox"/> Éditer Copier Supprimer	4	Café BIO	0	0
<input type="checkbox"/> Éditer Copier Supprimer	5	Chocolat chaud	0	0
<input type="checkbox"/> Éditer Copier Supprimer	6	Thé	0	0

On a bien une mise à jour qui a été effectuée dans la base de données. Ainsi si je rachète un café ou cappuccino, ceux-ci vont s'ajouter.

Quatrième version - Machine à café finale et sécurisée:

Sur cette version, on arrive à la concrétisation de notre machine à café. Elle est dynamique et vérifie si le solde de la carte correspond au solde dans la base de donnée et si c'est le cas, elle débite sur la carte et sur la bdd. Donc cela permet de rediriger l'étudiant vers la borne de recharge et mettre à jour son solde si le solde de la carte n'est pas bon et ainsi éviter des ajouts/rejouent.

PARTIE RESULTAT CAFEDELIKA :

Solde d'exemple :
Sur la BDD :

↔T↔	▼ boisson_id	boisson_nom	nombres_ventes ▼	1	montant_total
□ Éditer				1	0.2
📋 Copier					
🗑 Supprimer					
	1	Café			

MAJ 23/11/2023 : Mise à jour du nombre d'octets: 2 à 4 octets

Suite au passage de 2 à 4 octets sur notre carte à puce, il fallait apporter quelques correctifs afin de garantir le bon fonctionnement de la lecture de la carte, notamment sur le solde qui allait être mis sur 4 octets.

Huitième partie - Vulnérabilités :

Rubrovitamin.c :

1. Introduction

1.1 Objectif du Programme

Le programme "Rubrovitamin" vise à mettre en œuvre une carte à puce pour étudiant et ainsi, avoir une gestion de paiements.

1.2 Composants Principaux

- Programme "Rubrovitamin.c"
- Applications Associées : Lubiana.py, Cafedelika.py, Berlicum.py

2. Fonctionnement Général

2.1 Structure du Programme

Le programme est divisé en deux classes principales :

- **Classe de Personnalisation (0x81)** : Gestion des données personnelles.
- **Classe de Gestion de Paiements (0x82)** : Gestion du solde et des transactions.

2.2 Initialisation (ATR)

La procédure Answer To Reset (ATR) initialise la communication avec le système d'accès contrôlé en définissant le protocole et en envoyant des informations sur la carte.

3. Relations et Modes de Communication

3.1 Communication Interne

Les applications communiquent via des commandes APDU (Application Protocol Data Units).

3.2 Relations avec d'Autres Applications

- **Lubiana.py** : Envoie directement des instructions via une interface python.
- **Cafedelika.py** : Machine à café qui utilise une base de données pour gérer le solde et le transmet/vérifie à la carte à puce.

4. Vulnérabilités et Dysfonctionnements

4.1 Vulnérabilités Connues

1. Envoie d'instructions directement dans la carte via scat.

5. Solutions et Mécanismes de Sécurité

5.1 Solutions aux Vulnérabilités

1. Chiffrer la carte ?
2. Bloquer la carte et seul le logiciel lubiana à le droit d'accéder à la carte de l'étudiant.

[Lubiana.py](#) :

Présentation générale de Lubiana :

Notre programme `lubiana.py` est un logiciel administrateur destiné à interagir avec une carte à puce, la carte étudiante. Voici un résumé de ses principales fonctionnalités :

1. **Initialisation de la carte** (``init_smart_card``): Cette fonction initialise la connexion avec le lecteur de carte et vérifie la présence d'une carte. Elle stocke également l'ATR (Answer to Reset) de la carte, qui est essentiel pour la communication ultérieure.

2. **Fonctions d'impression** (``print_version``, ``print_nom``, ``print_prenom``, ``print_birth``, ``print_etu``, ``print_solde``): Ces fonctions envoient des commandes à la carte à puce et affichent les données renvoyées. Par exemple, elles peuvent afficher la version de la carte, le nom, le prénom, la date de naissance, le numéro étudiant et le solde.

3. **Fonctions d'écriture** (``intro_nom``, ``intro_prenom``, ``intro_birth``, ``intro_etu``, ``intro_credit``): Ces fonctions permettent d'écrire des informations sur la carte. Par exemple, l'introduction du nom, du prénom, de la date de naissance, du numéro étudiant et d'un crédit initial.

4. **Fonction pour réinitialiser le solde** (``reinit_solde``): Cette fonction réinitialise le solde de la carte à zéro.

5. **Menu principal** (``print_menu``, ``main``): Ces fonctions gèrent l'affichage du menu principal et permettent à l'utilisateur d'interagir avec le logiciel en choisissant différentes options.

Relations avec les autres applications et modes de communication :

Le code actuel de ``lubiana`` est un logiciel indépendant qui interagit directement avec une carte à puce. Elle ne communique pas avec la BDD.

Vulnérabilités potentielles:

1. **Communication non sécurisée** : Les données entre le logiciel et la carte ne sont pas sécurisées, cela pourrait entraîner des vulnérabilités.

2. **Injection de commandes** : Les commandes APDU sont construites à partir d'entrées utilisateur sans validation appropriée. Cela pourrait conduire à des attaques par injection de commandes via SCAT par exemple.

Propositions de solutions:

1. **Contrôle d'accès insuffisant** : Il faut s'assurer que seules les personnes autorisées peuvent accéder aux fonctionnalités sensibles de la carte.

2. **Stockage sécurisé des données** : Si des informations sensibles sont stockées, il faudrait correctement les chiffrées et protégées contre l'accès non autorisé.

[Cafédelika.py](#) :

Fonctionnement Général

Initialisation de la Carte à Puce (smart card) :

- Le script utilise la bibliothèque `smartcard` pour communiquer avec la carte à puce.
- La fonction `init_smart_card()` initialise la connexion avec le lecteur de carte et obtient l'ATR (Answer to Reset) de la carte.

Impression de l'État de la Carte :

- La fonction `print_solde()` envoie une APDU (Application Protocol Data Unit) pour obtenir le solde de la carte.
- Les fonctions `print_nom()` et `print_prenom()` font de même pour récupérer le nom et le prénom stockés sur la carte.

Enregistrement des Transactions :

- La fonction `enregistrer_transaction()` enregistre les transactions dans une base de données MySQL locale.
- Elle débite le solde de l'étudiant, met à jour les statistiques de vente pour les boissons, et commit les transactions dans la base de données.

Débit de la Carte et Enregistrement de Transaction :

- La fonction `debiter_carte()` débite le montant de la carte après avoir obtenu le solde actuel.
- Elle vérifie le solde et enregistre la transaction dans la base de données.

Menu Principal :

- La fonction `print_menu()` affiche un menu pour choisir une boisson.
- La fonction `main()` gère la boucle principale du programme, permettant à l'utilisateur de choisir des boissons jusqu'à ce qu'il choisisse de quitter.

Relations avec d'Autres Applications / Modes de Communication

Base de Données MySQL :

- Le script interagit avec une base de données MySQL pour enregistrer les transactions, récupérer le solde des étudiants, etc.
- Il utilise la bibliothèque `mysql.connector` pour se connecter à la base de données.

Vulnérabilités, Dysfonctionnements ou Attaques Possibles

Manque de Sécurité de Communication :

- Les communications avec la carte à puce ne sont pas chiffrées, ce qui peut être un risque de sécurité.

Gestion d'Erreurs Limitée :

- La gestion d'erreurs pourrait être améliorée pour fournir des informations plus précises en cas de problème.

Protection des Informations Sensibles :

- Les informations sensibles telles que les détails des cartes, noms, etc.. devraient être protégées.

Injection SQL Potentielle :

- Les requêtes SQL dans le script pourraient être vulnérables aux attaques par injection SQL. L'utilisation de paramètres de requête devrait être envisagée.

Solutions Proposées

Chiffrement des Communications :

- Utiliser des protocoles de chiffrement pour sécuriser les communications entre le script et la carte à puce.

Amélioration de la Gestion d'Erreurs :

- Ajouter des mécanismes de gestion d'erreurs plus robustes pour fournir des informations claires en cas de problème.

Protection des Informations Sensibles :

- Utiliser des mécanismes de protection appropriés (hashage, chiffrement) pour les données sensibles stockées et transmises.

[Berlicum.py](#) :

Fonctionnement Général

1. `init_smart_card()`

Cette fonction initialise la connexion avec un lecteur de cartes à puce. Elle liste les lecteurs disponibles, se connecte au premier lecteur, et stocke l'ATR (Answer To

Reset, une réponse standard d'une carte à puce lorsqu'elle est réinitialisée) de la carte insérée.

2. print hello message() Affiche un message de bienvenue pour le logiciel embarqué sur la borne de recharge.

3. print menu() Affiche un menu d'options pour l'utilisateur.

4. print nom(), print prenom(), print birth(), print étu()

Ces fonctions récupèrent et affichent différentes informations personnelles stockées sur la carte à puce, comme le nom, le prénom, la date de naissance, et le numéro d'étudiant.

5. print solde() Affiche le solde actuel sur la carte à puce.

6. afficher informations() Regroupe et affiche les informations personnelles de l'utilisateur (nom, prénom, date de naissance).

7. lire numero etudiant(): Lit et renvoie le numéro d'étudiant stocké sur la carte à puce.

8. consulter bonus() : Consulte la base de données pour obtenir et afficher les bonus associés à un numéro d'étudiant.

9. lire solde carte() : Lit et renvoie le solde disponible sur la carte à puce.

10. écrire solde carte (montant) : Met à jour le solde sur la carte à puce avec un nouveau montant.

11. transférer bonus() : Transfère un montant de bonus de la base de données vers la carte à puce.

12. consulter crédit() : Affiche le crédit disponible sur la carte.

13. lire solde db(etu_num) :Lit le solde de bonus dans la base de données pour un numéro d'étudiant donné.

14. recharger carte() :Permet à l'utilisateur de recharger sa carte avec un montant spécifié.

15. main() : La fonction principale qui initie le programme, affiche le menu et gère les interactions de l'utilisateur.

•

Relations avec d'Autres Applications / Modes de Communication

Base de Données MySQL :

- Le script interagit avec une base de données MySQL pour enregistrer les transactions, récupérer le solde des étudiants, etc.

- Il utilise la bibliothèque `mysql.connector` pour se connecter à la base de données.

Vulnérabilités, Dysfonctionnements ou Attaques Possibles

Attaque par Injection SQL :

- Un attaquant pourrait exploiter des failles d'injection SQL pour accéder ou modifier la base de données.
- de plus, le stockage en clair de la base de donnée permettrait de se connecter dans le réseau local

Absence de code pin :

- en cas de vol, le voleur a un accès au crédit de la personne possédant une carte.

Solutions Proposées

Mettre en place une validation stricte des entrées utilisateur pour éviter les injections SQL et autres formes de manipulations malveillantes.

Utiliser le cryptage pour protéger les données sensibles, en particulier lors de la communication avec la base de donnée.

Ajouter un code pin/puk

Méthode de sécurité pour la carte :

- Anti-arrachement (obligatoire pour ce projet) : si un arrachement de carte est survenu lors d'une transaction, il faut remettre la carte dans un état correcte lors de la prochaine utilisation de la carte.

- Anti-rejoue (facultatif pour ce projet) : Grâce à notre système vérifiant l'état de la base de donnée, les étudiants malins ne pourront pas s'ajouter des crédits sur la carte en utilisant les APDU à leur escient. De ce fait, si la carte n'a pas le même montant que la BDD, alors il ne pourra pas s'acheter de café et devra aller à la borne de recharge pour mettre à jour son solde personnel sur la carte. Donc, la BDD sera toujours considérée comme vraie. Le problème qu'il pourrait y avoir, c'est qu'il accède à notre BDD en tant que root et modifie lui même les valeurs, mais c'est un autre problème de sécurité.

Conclusion :

En conclusion, le projet "La Carotte Électronique" a abouti à la création d'un système novateur de récompenses pour les étudiants de l'IUT de Vélizy. À travers la conception et l'implémentation de plusieurs composants logiciels, notre équipe a réussi à mettre en place un écosystème complet, intégrant des cartes à puce, un logiciel de personnalisation, une base de données, un système de gestion des récompenses, et une borne de recharge. Ce projet offre non seulement une solution pratique pour stimuler l'assiduité et la performance académique des étudiants, mais il introduit également une approche technologique moderne dans le quotidien du campus.

La collaboration étroite entre les membres de l'équipe a permis de surmonter des défis complexes, de la conception des fonctionnalités des cartes à puce à la mise en place d'un système de gestion de base de données robuste. L'utilisation de noms inspirés de variétés de carottes pour les différents logiciels ajoute une touche esthétique à l'ensemble du projet.

En outre, le respect des bonnes pratiques de développement logiciel, l'utilisation du système de gestion de versions GIT, et la gestion d'équipes ont renforcé l'efficacité et la cohérence du processus de développement.