Jean-Roch BECART
~~Mathieu PANIEZ~~

Institut Universitaire Technologique
Informatique d'AMIENS

Seconde année, GROUPE SI

Tuteur : M. DUPUIS

Bilan Projet Tuteuré

Secure Windows Logon
Authentification par carte à puce sur station de travail Windows
Jean-Roch BECART et ~~Mathieu PANIEZ~~

05/06/2003

A l'intention d'Arnaud DUPUIS

SWL
Secure Windows Logon

I. Introduction.

Notre projet consiste à sécuriser l'accès à un ordinateur personnel relié à un réseau ou non, de façon sécurisée. L'utilisateur n'a plus de mot de passe à taper ni de login, il lui suffit d'entrer une carte à puce dans un lecteur pouvant être placé dans une baie 3,5 pouce ou à l 'arrière de l'ordinateur sur le port série (ou USB suivant matériel). Il entre juste un code pin (optionnel) pour prouver qu'il est bien le propriétaire de cette carte à puce, comme cela se fait pour les cartes bancaires, hormis que notre système est beaucoup plus sécurisé sans vouloir remettre en cause ces types de carte.

Ce programme permet donc de remplacer les contrôles d'accès standard à un PC par le biais d'une carte à puce. Cette carte pourrait aussi être utilisée pour contrôler et gérer un nombre important de paramètres et d'applications, tel que la gestion d'un quota d'impression, l'ouverture d'une porte et pourquoi pas l'accès à un parking ou un self service.

II. Analyse du problème et choix de solution.

Plusieurs façons existent pour se loguer, nous avons du faire un choix entre :

° Utiliser le système d'authentification Windows Kerberos V5, développé au MIT (Boston) mais peu documenté.

° Lancer notre programme au moment du Login de Windows et remplir les champs (Login/mot de passe/domaine, etc.…) et laisser Windows lancer la procédure habituelle.

° Faire notre propre interface de Login qui vérifiera d'elle-même l'authenticité de la carte et du porteur.

Nous avons donc choisi la deuxième méthode qui consiste à refaire la dll Msgina qui permet d'avoir une interface graphique lors du login sur une station windows.

Nous avons préféré ce choix par souci de sécurité, de portabilité (Windows 98, Windows NT, Windows 2000 et XP).

Nous avons utilisé, pour le développement, Visual C++, qui permet une réalisation intuitive de fichier DLL et aussi de modifier les projets du code source, libre pGina, développés en VC++.

III. Explication sur la procédure d'authentification.

Ps=Programme coté serveur = Station de travail
Pc=Programme cote client = carte à puce

Pour l'authentification sur un système Windows, notre programme réalise le dialogue suivant :

Au démarrage de Windows une fenêtre demande l'entrée de la carte à puce. A l'insertion de la carte dans le lecteur, une nouvelle fenêtre apparaît, elle invite l'utilisateur à entrer son code pin, il valide par OK et il est autorisé à utiliser windows.

Explications :

Action	PC→CARTE	CARTE→PC
Phase d'authentification du Ps(pour éviter un cassage de la clé contenu dans la carte, ce qui serait déjà une véritable prouesse)		
Insertion carte lecteur	Ps envoie demande à la carte.	
		Pc répond en envoyant nombre aléatoire.
	Ps crypte nombre avec clé.	
		Pc vérifie et répond 9000 si ok Sinon répond 90FF.1er verrou débloquer.
	Ps demande nom login	
		Carte envoie réponse XXX
Phase d'authentification du porteur de la carte (si une carte est volée, elle ne peut pas être utilisée), cette phase peut s'avérer contraignante c'est pourquoi, nous avons prévu une option pour ne pas toujours l'utiliser.		
Une fenêtre invitant à taper un code pin apparaît. « Veuillez saisir votre code pin M. XXX »	Ps envoi le code pin crypté à Pc	
		Pc décrypte et vérifie code si code bon, étape suivante et mise du compteur d'essai à zéro, ouverture du second verrou. Sinon incrémentation du compteur d'essai et affichage d'une fenêtre code erroné puis relance de la procédure.
Phase d'authentification de la carte par Ps.		
	Ps envoie code aléatoire à la carte.	
		Pc crypte données et renvoie résultat.
	Ps vérifie et si ok demande les infos suivantes à la carte : Login Domaine + autre données en fonction des besoins ou de la demande.	
		Pc crypte données et renvoie résultat.

Windows peut maintenant finir la phase de login en vérifiant de son coté le mot de passe qui lui a été transmis.

IV. Phase d' extraction de la carte.

(optionnel car ceci est contraignant pour un usage comme celui de l'IUT, cas ou qq'un part)
Quand la carte est enlevée :

- L'écran de veille démarre / ou la station est bloquée.
- Il faut réintroduire la carte pour débloquer la station.

Le PC est donc sécurisé et bloqué jusqu'au moment où la carte est remise dans le lecteur.

Pour se déconnecter de la session ouverte l'utilisateur fait comme d'habitude, la démarche est la même. Il peut ensuite retirer sa carte du lecteur.

V. Détails des méthodes et explication du programme.

Fonctionnement sommaire d'une carte à puce.

La norme actuelle est iso 7816-3, le protocole est T=1(pour la enhanced basicCard)
Une carte à puce a un fonctionnement passif, elle ne répond qu'à une question posée. Tous les échanges de données se font donc sous la forme de challenge question(s)/réponse(s).

Pour établir un dialogue avec une carte à puce, le lecteur envoie un reset à la carte qui répond par un ATR, cet ATR est une chaîne de données qui définit les tensions de fonctionnements de la carte, son protocole de communication ainsi que d'autres données utiles au dialogue.

La suite du dialogue commence par 5 groupes de 2 octets, qui sont envoyés à la carte, qui se définissent ainsi :

CLA / INS / P1 / P2 / LEN

CLA : Lors d'une communication avec une carte on défini une classe sur 2 octets qui correspond à notre application de carte à puce.
INS : C'est une commande. (Ex : dir, copy, delete, sous dos)
P1 et P2 sont des paramètres.
LEN correspond à la longueur des données que l'on doit, soit recevoir, soit envoyer au lecteur.

Après un échange de données la carte répond (bien souvent) par 90 00 pour confirmer que le dialogue c'est bien passé, sinon elle peut répondre par un code d'erreur 6F XX (définit par le concepteur)

Programme réalisé :

Plusieurs fichiers ont été développés ou modifiés dans ce projet.
Il faut distinguer les programmes créés :

- ceux de la carte à puce (fait en langage basicCard)
- ceux permettant une gestion de la carte (2 exécutables)
- ceux utilisés par le système pour se loguer sous Windows (2 dll modifiés plus 1 créée)

Il faut installer pGina, remplacer le fichier Dummyplugin.dll (le notre) dans le répertoire du plugins de pGina et configurer pGina.
Ensuite il reste à copier le fichier projet.dll dans le répertoire WINNT\SYSTEM32.
Ce sont les seuls fichiers utilisés par Windows pour se loguer.

Les deux autres exécutables sont des programmes de gestion de la carte.

Ne pas oublier que le quota d'impression est lisible par l'intermédiaire du balance reader.

VI. Sources Commentées.

```
Rem    Programme de la basicCard.
Rem
Rem     IUT INFORMATIQUE AMIENS
Rem
Rem        Jean-Roch BECART
Rem   (option Système Industriel)
Rem
Rem  Projet tuteuré.(Arnaud DUPUIS)
Rem
Rem
Rem  Carte d'authentification, gestion d'impression, parking, self service...
Rem
Rem  La carte à deux clés: KEY 00 (clé du distributeur) and KEY 01 (clé de l'administrateur).
Rem  Pour un maximum de sécurité, toutes les commandes sauf GetCardData doivent
Rem  être cryptées avec l'algorithme &H12
Rem
Rem  Les commandes implémentées sont les suivantes:
Rem
Rem  PersonaliseCard (Amount As Long, PIN As String*4, Customer$, Passwd$)
Rem      Require la clé du distributeur. Une carte peut être personnalisée plusieurs fois
Rem
Rem  VerifyPIN (TestPIN As String*4)
Rem      Cette commande est appelée par les 3 commandes suivantes. Cette commande
Rem      reste valide jusqu'a ce que la carte soit éteinte ou mise à zéro.
Rem      Require administrateur Key ou distributeur Key.
Rem
Rem  IncreaseAmount (Diff As Long)
Rem      Ajoute un quota d'impression sur la carte. Require VerifyPIN.
Rem
Rem  DecreaseAmount (Diff As Long)
Rem      Enlève un quota sur la carte. Require VerifyPIN.
Rem
Rem  ChangePIN (NewPIN As String*4)
Rem      Change le code PIN. Require VerifyPIN.
Rem
Rem  GetCardData (Amount As Long, PINCount As Byte, Customer$)
Rem      Retourne les données de la carte. PINCount@ est le nombre de tentatives restantes
Rem      avant que la carte soit bloquée.
Rem      (Le nombre de tentatives alloué est défini par MaxPINErrors.)
Rem
Rem  GetPassword(pass$)
Rem      Retourne le password utilisateur
Rem
Rem

Option Explicit ' Empêche l'utilisation de variables indéfinies

#Include DEBITCRD.DEF   'Déclarations commune à la BasicCard et au programme.
#Include DEALER.KEY     'clé du distributeur
#Include ISSUER.KEY     'clé administrateur
#Include COMMANDS.DEF
#include PReader.def

Declare ApplicationID = ApplicationName$
Disable Encryption &H11

Rem  ----données permanentes----
 Eeprom Personalised = False
 Eeprom Balance As Long
 Eeprom PIN As String*4
```

```
 'Changer la ligne suivante pour régler le master PIN initial
 Eeprom MasterPIN As String*6 = "123456"
 Eeprom CustomerName$
 Eeprom WndPass$
 Eeprom PINErrors

Rem On s'assure que le balance reader aura toujours accès
 Eeprom ShadowBalance As Long
 Eeprom Committed = False

Rem  Public data (re-initialised when card is reset)
 Public PINVerified = False
 Public MasterPINVerified = False

Rem  Subroutine declaration
 Declare Sub ChangeBalance (NewBalance As Long)
 Declare Sub CheckAlgorithm()

Rem Si on a des problèmes de connexion
 If Committed Then
   Balance = ShadowBalance
   Committed = False
 End If


Rem
Rem  Commandes de la carte
Rem


Command &H80 &H01 SetPasswd (Passwd$)
  Call CheckAlgorithm()
  If KeyNumber <> 0 Then SW1SW2 = swIssuingKeyRequired : Exit
  WndPass$ = Passwd$
End Command

Command &H80 &H00 PersonaliseCard (Amount As Long, NewPIN As String*4, Name$)
  Call CheckAlgorithm()
  If KeyNumber <> 0 Then SW1SW2 = swIssuingKeyRequired : Exit
  Personalised = False ' In case power is lost in the next three statements
  Balance = Amount
  CustomerName$ = Name$
  PIN = NewPIN
  Personalised = True
End Command

Command &H80 &H02 VerifyPIN (TestPIN As String*4)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
  If KeyNumber <> 0 And PINErrors > MaxPINErrors Then _
    SW1SW2 = swPINErrorsExceeded : Exit
  If TestPIN = PIN Then PINErrors = 0 : PINVerified = True : Exit
  PINErrors = PINErrors + 1
  SW1SW2 = swInvalidPIN
  PINVerified = False
End Command

Command &H80 &H0F GetPassword(pass$)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
        rem//on vérifie ici si l'utilisateur a donné son code pin
        rem//sinon le password n'est pas envoyé
  If Not (PINVerified OR MasterPINVerified) Then SW1SW2 = swPINRequired : Exit
  pass$=WndPass$
End Command

Command &H80 &H04 IncreaseAmount (Diff As Long)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
  If Not (PINVerified OR MasterPINVerified) Then SW1SW2 = swPINRequired : Exit
  Call ChangeBalance (Balance + Diff)
End Command

Command &H80 &H06 DecreaseAmount (Diff As Long)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
  If Not (PINVerified OR MasterPINVerified) Then SW1SW2 = swPINRequired : Exit
```

```
  If Diff > Balance Then SW1SW2 = swInsufficientFunds : Exit
  Call ChangeBalance (Balance - Diff)
End Command

Command &H80 &H08 ChangePIN (NewPIN As String*4)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
  If Not (PINVerified OR MasterPINVerified) Then SW1SW2 = swPINRequired : Exit
  PIN = NewPIN
End Command

Command &H80 &H0A GetCardData (Amount As Long, PINCount, Name$)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Amount = Balance
  PINCount = MaxPINErrors - PINErrors
  Name$ = CustomerName$
End Command

Rem  Sub CheckAlgorithm()
Rem     Vérifie que l'algorithme de cryptage approprié est valide(&H12
Rem     dans la basicCard Compact, &H21 dans la enanhced)

Sub CheckAlgorithm()
#IfDef CompactBasicCard
  If Algorithm <> &H12 Then SW1SW2 = swEncryptionRequired : Exit
#ElseIfDef EnhancedBasicCard
  If Algorithm <> &H21 Then SW1SW2 = swEncryptionRequired : Exit
#Else
  If Algorithm <> &H23 Then SW1SW2 = swEncryptionRequired : Exit
#EndIf
End Sub

Rem changement de variable dans le cas ou le Balance reader
Rem tomberait en panne

Sub ChangeBalance (NewBalance As Long)
  ShadowBalance = NewBalance
  Committed = True
  Balance = ShadowBalance
  Committed = False
End Sub

Command &H80 &H0E ChangeMasterPIN (NewPIN As String*6)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
  If Not MasterPINVerified Then SW1SW2 = swPINRequired : Exit
  MasterPIN = NewPIN
End Command

Command &H80 &H0C VerifyMasterPIN (TestPIN As String*6)
  If Not Personalised Then SW1SW2 = swNotPersonalised : Exit
  Call CheckAlgorithm()
  If KeyNumber <> 0 Then _
    SW1SW2 = swIssuingKeyRequired : Exit
  If TestPIN = MasterPIN Then MasterPINVerified = True : Exit
  SW1SW2 = swInvalidPIN
  MasterPINVerified = False
End Command

Rem Affichage sur l'afficheur portable à cristaux liquide

Command &HC8 &H00 PRDisplay(RecordNumber as Byte, DataFormat as Byte, DigitCount as Byte, _
                            DecimalPoint as Byte, Delay as Byte, MoreData as Byte, _
                            Data as String)
  Dim BalanceData as Long
  Dim BalanceDataStr as String*4 at BalanceData
  select case RecordNumber
     case 0
       DataFormat = PRAlpha ' alphanumeric
       DigitCount=0         ' show all digits
       DecimalPoint=3       ' decimal point at 3 character (2 digits follow point)
       Delay=0              ' show till card is removed
       MoreData=PRNoMoreData ' no more data to show
       BalanceData=Balance  ' convert long to string
       Data="RESTE " + str$(Balance)  ' and send to balance reader
     case else
       DataFormat=PRAlpha
```

```
        DigitCount=0
        DecimalPoint=0
        Delay=1000 / PRDelayUnits ' 1 second to show
        MoreData=PRNoMoreData
        Data="ERR" ' Error message
    end select
End Command
```

Fonctions implémentées dans les divers Exe et dlls :

Pour des questions de visibilité nous avons préféré les laisser accessible par PC.

VII. Conseil pour que le système de sécurisation mis en oeuvre soit sans faille.

Conseils : notre système est efficace, mais cependant il ne faut pas négliger certaines règles de sécurité, voici ce qui doit être vérifié (ces règles sont vérifiées sur le réseau de l'IUT)

-Mot de passe dans le Bios pour empêcher quiconque de choisir le support de Boot. (Mettre C en premier sans quoi cela est inutile.)
-Que Bootkeys=0 dans le fichier msdos.sys pour ne pas avoir de menu au démarrage (dans le dos).
-Bloquer en lecture et écriture les répertoires dont l'utilisateur normal n'as pas besoin.
-Bloquer l'accès à la base de registre.

VIII. Compatibilité, analyse coût et développement .

-Compatible Windows 95/98/Me/NT/2000.(mais ce système est plus orienté réseau)
-Utilise un lecteur de carte bon marché type Cybermouse(port série, usb, baie 5p1/4 ou 3p1/2)
-Cartes à puces BasicCard les meilleures cartes rapport qualité/Prix
-Le logiciel doit être installé sur chaque poste.

Un système comparable au notre est vendu dans le commerce entre 200 et 1000 € pour un nombre de cartes et licences allant d'une dizaine à une centaine, sans compter l'investissement du lecteur de carte. Soit un coût comprit entre 20 et 10 € par carte.(sans compter l'investissement du lecteur de carte)

Le coût d'une carte vierge pour une centaine d'unité est faible moins de 4 €, le rapport coûts/bénéfices est donc très intéressant.

Le développement de cette application n'est pas excessivement coûteuse en temps et en moyens, mais il est relativement compliqué dans la mesure ou les tests doivent être fait au démarrage de l'ordinateur il faut a chaque fois lancé la machine, si il y a un bug la relancer en mode sans échec, restaurer les bons fichiers , redémarrer pour corriger enfin redémarrer pour re-tester. La perte de temps correspond à plus d'un tiers du temps de développement total du projet réparti comme suit.

-essai

-programmation dll (erreur, recherche d'information, choix d'une solution)
-erreur de driver(attention à l'installation du lecteur)

IX Conclusion.

Notre interface pc/carte est facilement portable pour d'autre type d'authentification, biométrie, analyse rétinienne, clé USB. Il s'agit donc d'un programme professionnel, exploitable, commercialisable et modulable, prêt pour les plus fortes exigences en matière de sécurité. Avec un peu plus de temps nous aurions pu étendre notre projet à Linux, de plus en plus utilisé, et qui s'avère être un marché porteur et non négligeable pour le futur. Cependant il reste quelques routines de code à améliorer, surtout dans la partie VC++, quelques tests d'erreur à effectué et un control perpétuel de la présence de la carte dans le lecteur(dans certain cas d'utilisation)

Nous avons cependant remplis notre objectif qui était de rendre un système d'authentification opérationnel, qui est très complet et pouvant recevoir des évolutions futures pour d'autres types d'utilisations ou d'autres types d'authentifications.

Shéma de fonctionnement de pGina.

Captures d'écrans.