

Guide d'intégration



DINKEYPRO

Protection de logiciels en C# Microsoft sous Visual Studio avec DinkeyPRO/FD



Contact Commercial :
Tél. : 02 47 35 70 35
Email : com@aplika.fr

Contact Technique :
Tél. : 02 47 35 53 36
Email : support@aplika.fr

Version 1.00 du 10/12/2009



aplika

La Foltière - 37270 AZAY SUR CHER
Tél. 33(0)2 47 35 70 35 - Fax 33(0)2 47 35 70 25 - e-mail : aplika@aplika.fr

Guide d'intégration DinkeyPRO/FD C# MICROSOFT SOUS VISUAL STUDIO



1. Introduction	3
2. Exemple de test projet DIITest.sln	3
2.1. Ouverture du projet – Test simple de la protection avec la méthode API	3
2.2. Protection du module DPWIN32.DLL	5
2.3. Compilation du projet de test et essai de protection.....	7
3. Le module de protection Dinkey pour C# DPWIN32.DLL.....	8
3.1. Module dpwin32.dll	8
4. Adaptation du code exemple à votre projet	10
5. Déploiement de votre application	11

Guide d'intégration DinkeyPRO/FD C# MICROSOFT SOUS VISUAL STUDIO

1. Introduction

Afin de vous aider dans les premiers pas de la pose de protection sur vos logiciels C# MICROSOFT SOUS VISUAL STUDIO, vous trouverez dans ce guide de prise en main rapide l'essentiel des commandes et fonctionnalités pour la bonne utilisation des clés DinkeyPRO/FD.

Méthode utilisée : API avec appel de dpwin32.dll.

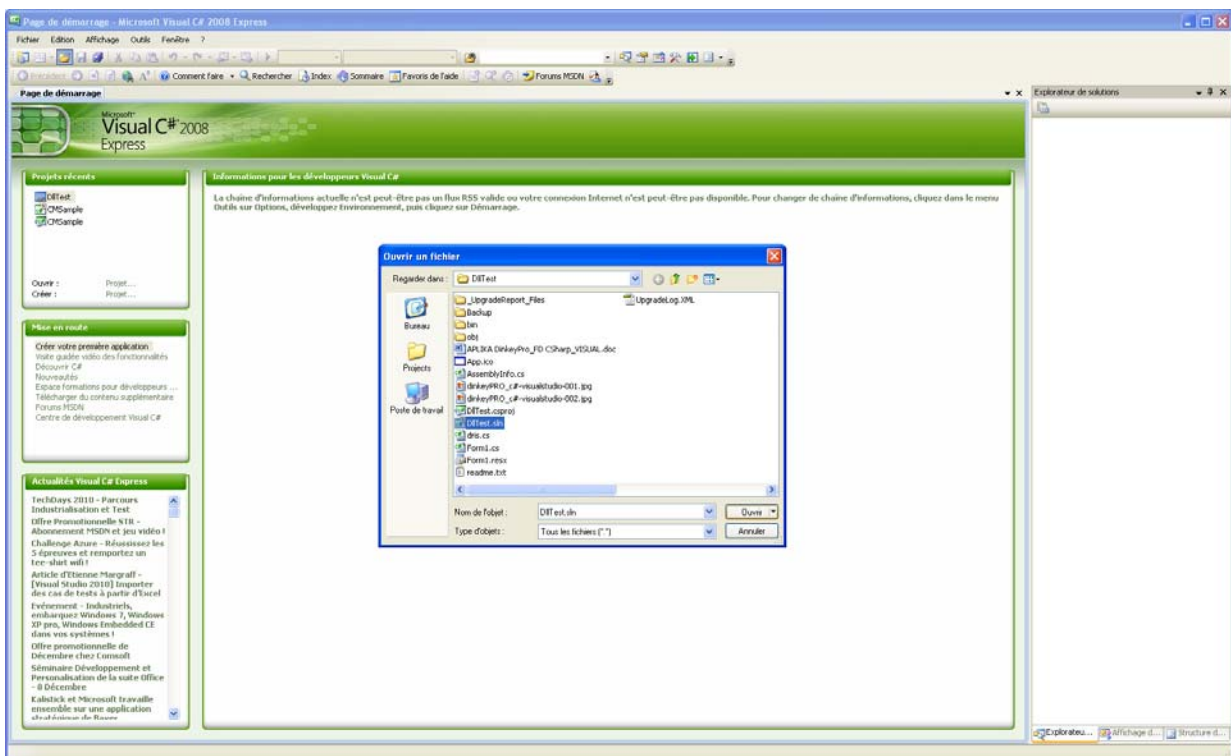
Le code C# MICROSOFT SOUS VISUAL STUDIO de cet exemple a été testé avec les versions 2003 et 2008 de C# MICROSOFT VISUAL STUDIO. Pour l'utilisation sous d'autres versions et/ou d'autres plateformes, nous vous conseillons d'effectuer systématiquement un test car nous ne pouvons garantir la compatibilité. En cas de difficulté nous contacter en prenant soin de nous communiquer la version de C# MICROSOFT VISUAL STUDIO utilisée.

Le fichier DllTest.dsw contient un exemple de code permettant l'appel à la protection Dinkey. Cet exemple a pour objectif de mettre en évidence le fonctionnement des APIs Dinkey afin de vous permettre une intégration facile dans vos projets C# MICROSOFT SOUS VISUAL STUDIO.

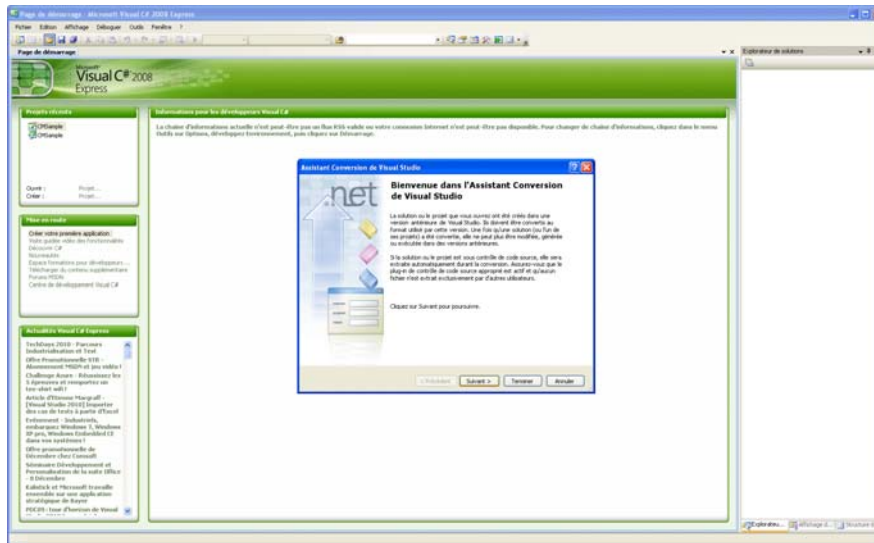
2. Exemple de test projet DllTest.sln

2.1. Ouverture du projet – Test simple de la protection avec la méthode API

- Ouvrez l'éditeur C# SOUS VISUAL STUDIO.
- Puis ouvrez la solution DllTest.sln qui contient un exemple simple de protection

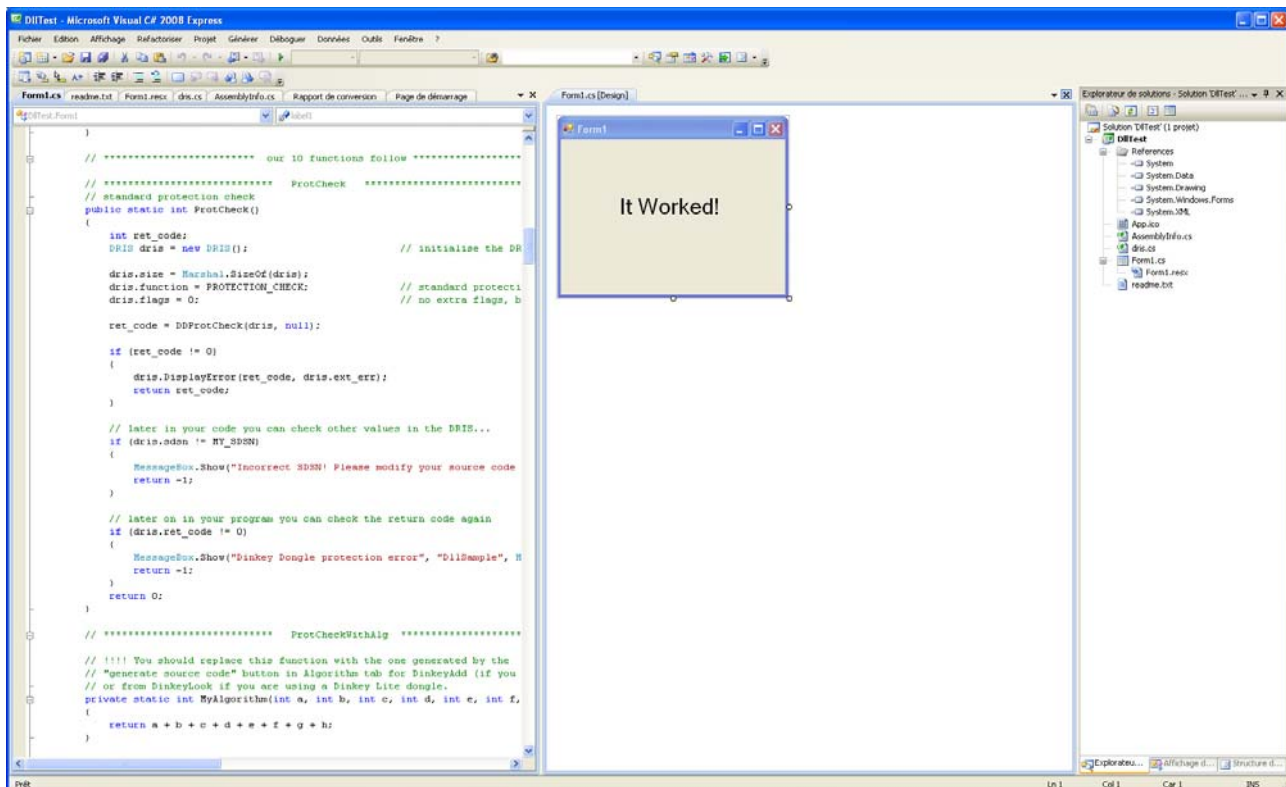


- En fonction de votre version de VISUAL STUDIO, une conversion peut être proposée.



- Le projet s'ouvre. Form1.cs dans la fenêtre code contient l'appel de l'API de dpwin32.dll et dris.cs contient la déclaration de la structure DRIS
- Modifier la ligne ci-dessous pour indiquer votre numéro de série développeur. Remplacer la valeur 10101 (valeur SDSN des clés de démonstration par la votre – Ce numéro vous est attribué lors de votre première commande de clés)

```
public const int MY_SDSN = 10101; // !!!! change this value to be
                                the value of your SDSN (demo = 10101)
```

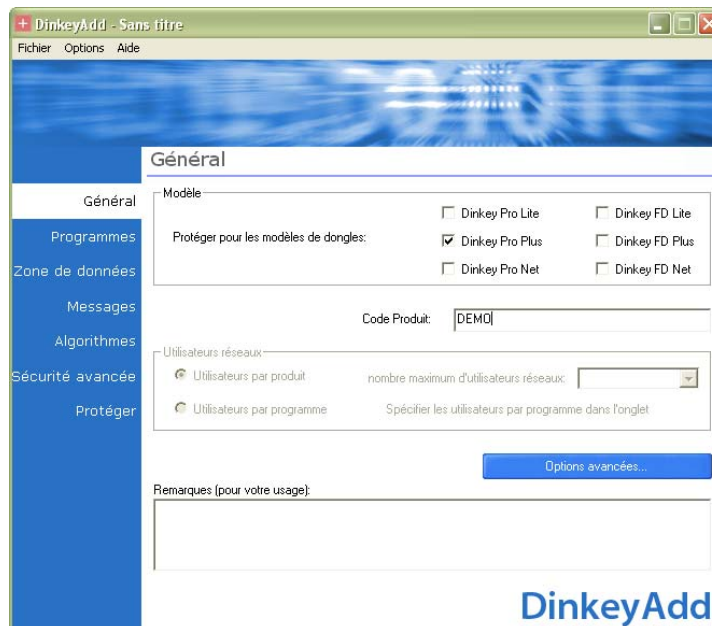


2.2. Protection du module DPWIN32.DLL

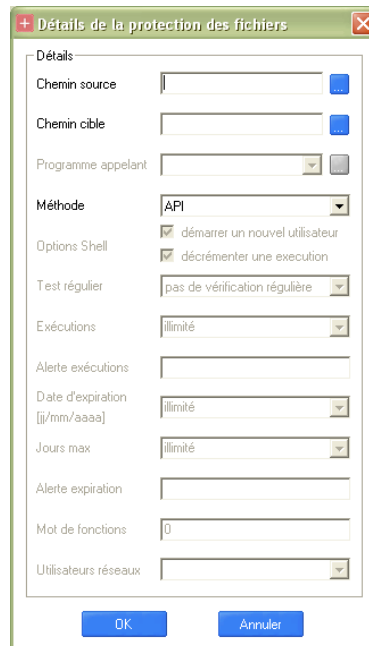
Ouvrez l'utilitaire DinkeyAdd.

Dans l'onglet "Général" :

- Précisez le type de clé que vous utilisez. En cas de doute, exécutez DinkeyLook afin d'afficher un diagnostic de la clé.
- Indiquez le Code Produit (pour les clés d'évaluation, celui-ci est "DEMO").



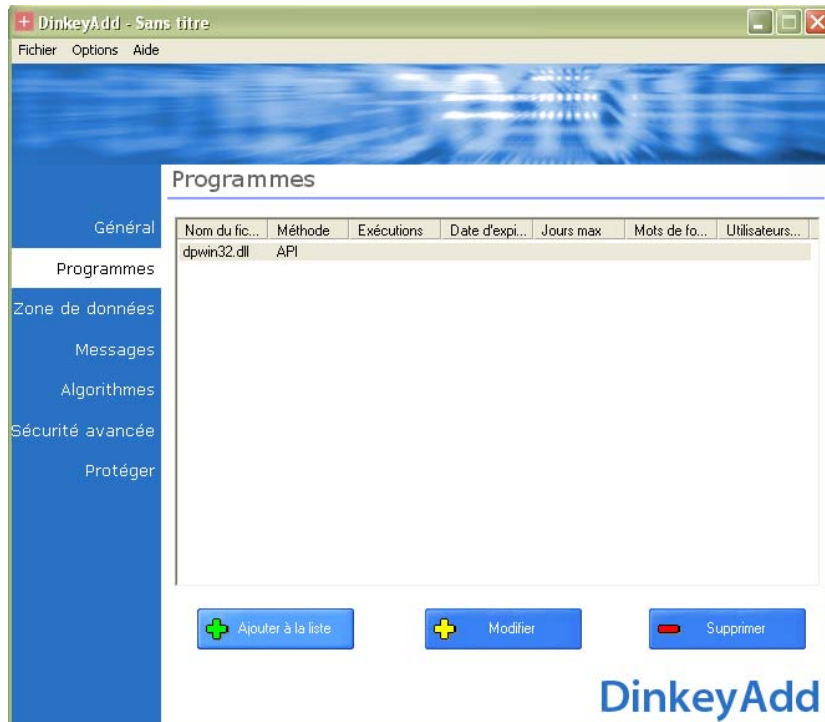
Dans l'onglet "Programmes" :



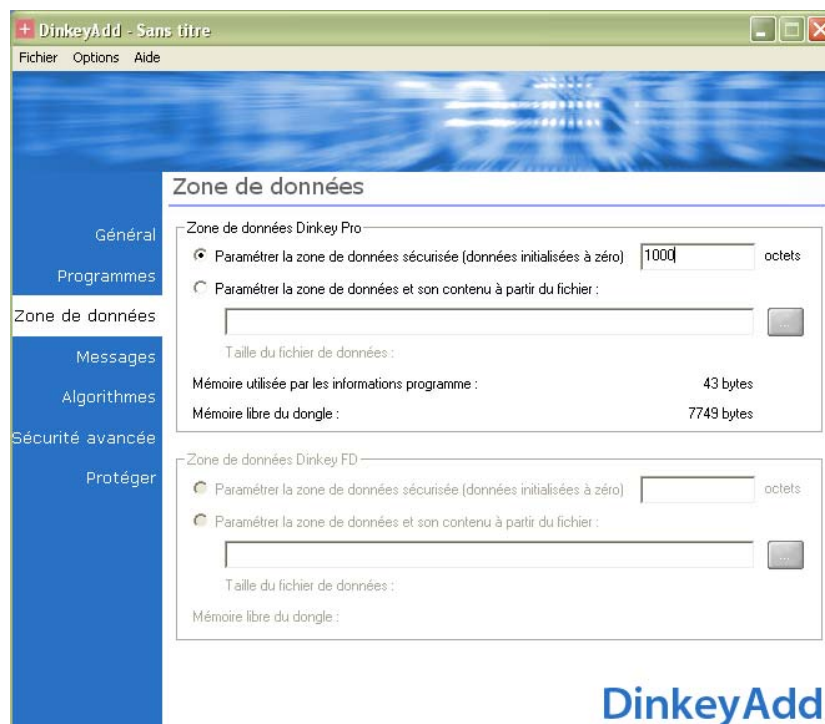
- Cliquez sur le bouton "Ajouter à la liste".

- Dans le champ "Chemin source", pointez sur le fichier dpwin32.dll (celui-ci se trouve par défaut dans le sous-dossier "Modules" du dossier d'installation de DinkeyPRO).
- Dans le champ "Chemin cible", indiquez l'emplacement vers lequel vous souhaitez qu'une copie protégée de dpwin32.dll soit placée (par exemple répertoire bin/release de votre projet).
- Vérifiez que la méthode "API" est bien sélectionnée.
- Laissez les valeurs par défaut pour les champs suivants.
- Validez en cliquant sur le bouton OK.

Le fichier dpwin32.dll est ajouté à la liste des programmes.



Dans l'onglet "Zone de données" :

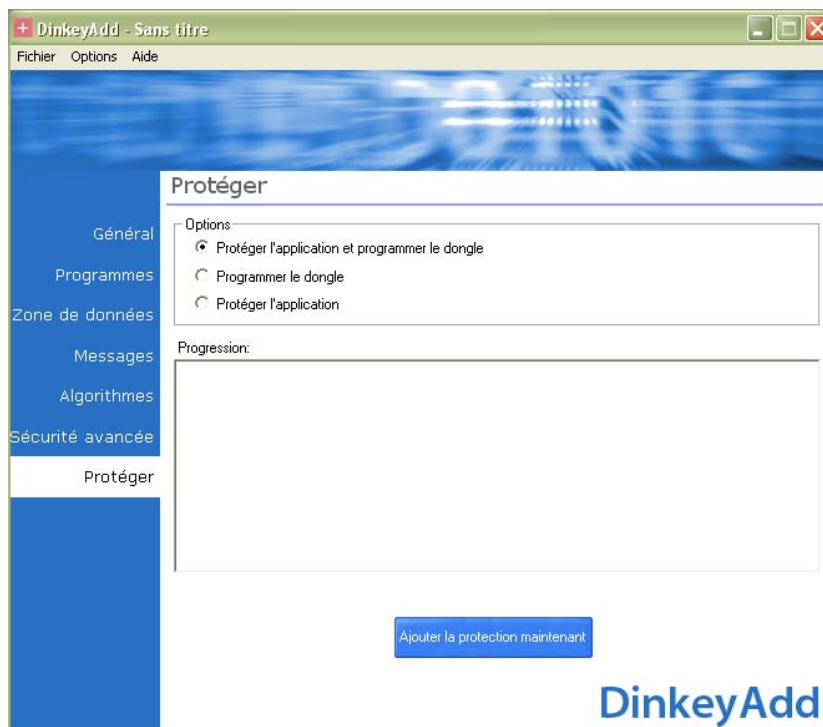


- Initialisez la taille de la zone de données sécurisée, par exemple 1000 octets. Le code exemple fourni permet de tester la lecture/écriture à partir de la zone de données sécurisée. Ceci ne fonctionnera que si cette zone est initialisée avec une taille suffisante pour recevoir les données de test.

Remarques :

- Selon le type de clé (DinkeyPRO ou DinkeyFD) choisi dans l'onglet "Général", les paramètres de la zone de données correspondante seront activés dans l'onglet "Zone de données".
- Seules les versions Plus et Net disposent d'une zone de données sécurisée.

Dans l'onglet "Protéger" :



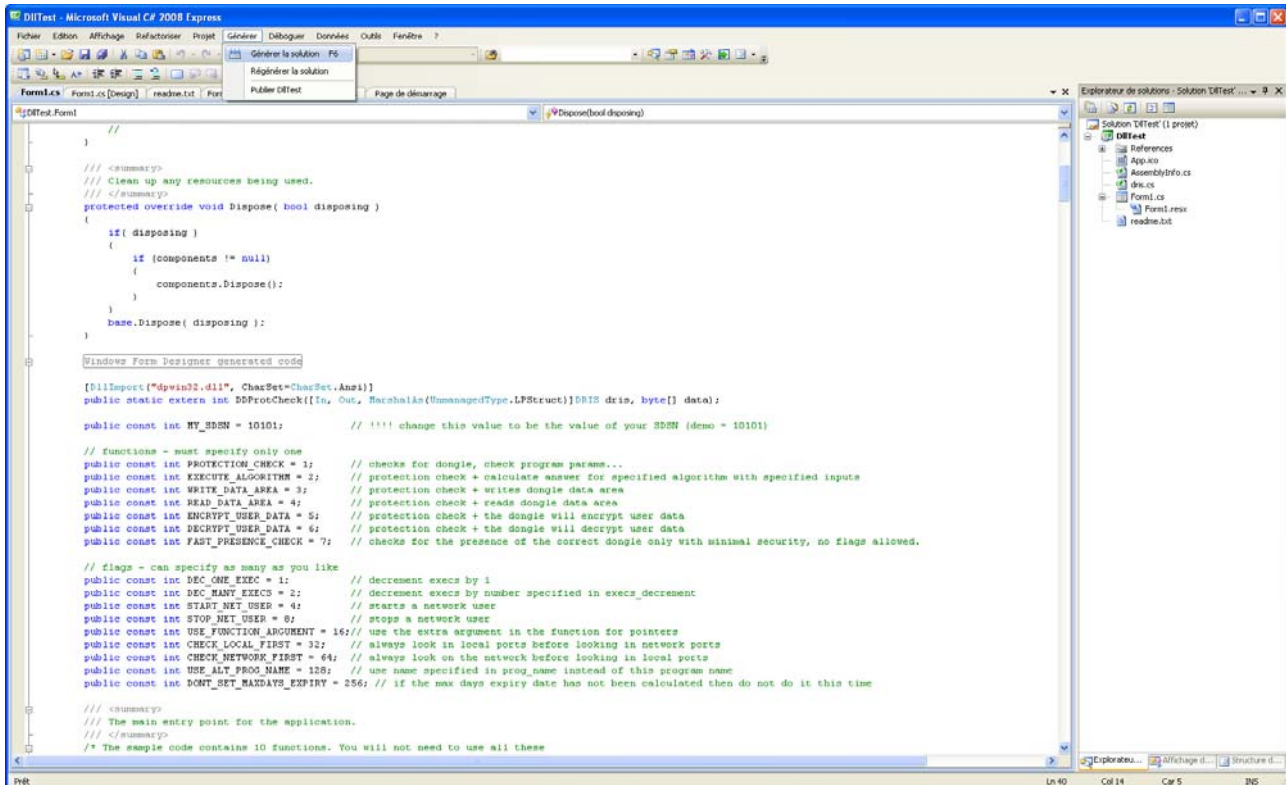
- Vérifiez que l'option "Protéger l'application et programmer le dongle" est sélectionnée.
- Assurez-vous que le dongle est connecté.
- Cliquez sur le bouton "Ajouter la protection maintenant"
- Un message de confirmation apparaît.

Vous disposerez alors d'un dongle correctement programmé, ainsi que d'une version protégée de dpwin32.dll dans le dossier vers lequel vous aurez pointé dans le champ "Chemin cible" de la fenêtre "Détails de la protection des fichiers".

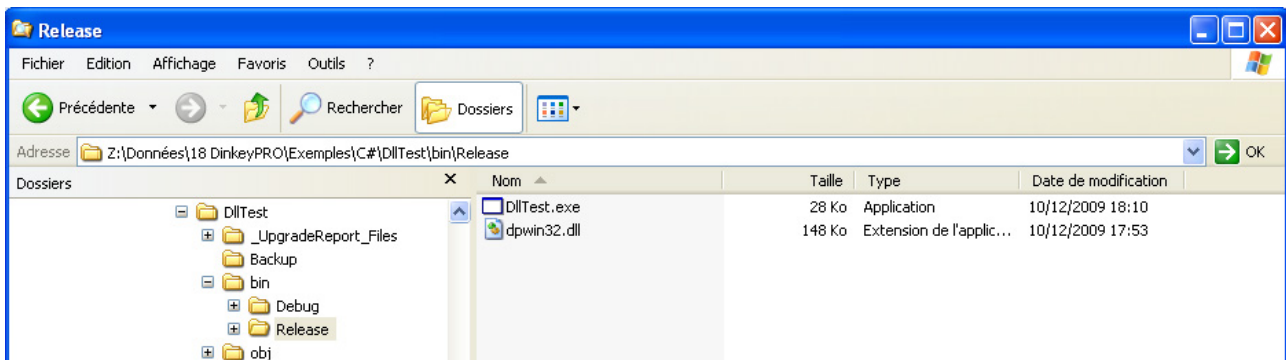
Copiez alors la DLL ainsi protégée dans le répertoire bin/release contenant votre projet de test VISUAL STUDIO.

2.3. Compilation du projet de test et essai de protection

Votre projet est ouvert et prêt à être compilé et la DLL dpwin32.dll est protégée. Vous pouvez donc compiler le projet afin de tester la protection en lançant l'exécutable protégé.



L'exécutable est alors généré et peut être lancé en dehors de l'environnement de VISUAL STUDIO à partir du répertoire Release ou directement à partir du débog C#. Si vous avez oublié de copier les fichiers obtenus à l'étape 'Protection du module DPWIN32.DLL' alors une erreur indiquant que dpwin32.dll n'est pas trouvée apparaîtra. Pour la résoudre copiez les fichiers pour obtenir la liste visible sur l'écran ci-dessous.



Lancez le programme DITest.exe avec le dongle connecté puis déconnecté.

Vous pouvez maintenant adapter cet exemple à votre besoin pour protéger vos EXE ou DLL C#. Notre équipe technique reste bien sûr à votre disposition. Nous détaillons ci-après les informations qui vous permettront de mieux comprendre les fonctions de dpwin32.dll utilisées dans notre exemple et vous invitons à consulter notre Manuel du programmeur. Vous trouverez également dans notre exemple 10 fonctions de test prêtes à l'emploi permettant d'utiliser toutes les capacités de nos différents modèles de clés.

3. Le module de protection Dinkey pour C# DPWIN32.DLL

3.1. Module dpwin32.dll

Le test de protection C# MICROSOFT SOUS VISUAL STUDIO est effectué en faisant appel à dpwin32.dll. Ce dernier permet d'établir le dialogue entre la clé Dinkey et votre programme via l'utilisation de dpwin32.dll. Vous devrez donc protéger cette DLL en utilisant la méthode de protection API comme décrit en section 3.2.

Le chemin de dpwin32.dll doit être indiqué. Si vous n'indiquez pas de chemin à C# Microsoft sous Visual Studio ; il ira alors par défaut chercher dpwin32.dll dans le répertoire contenant le programme protégé.

La DLL dpwin32.dll contient les fonctions de test de présence de la clé ainsi que les fonctions de lecture et écriture de données. Les entrées et sorties des fonctions sont décrites ci-après et leur lecture/écriture s'effectue par l'intermédiaire de la structure DRIS également décrite ci-après.

INPUTS	OUTPUTS	return
'in' cluster: function flags execs_dec rw_offset rw_length rw_data alt_prog_name var_a var_b var_c var_d var_e var_f var_g var_h alg_number	'out' cluster: ret_code ext_err type model sdsn prodcod dongle_number update_number data_area_size max_alg_num execs exp_day exp_month exp_year features net_users alg_answer fd_capacity fd_drive	'return' integer: Correspond à la valeur de retour de dpwin32.dll. Pour l'interprétation des valeurs se reporter à la documentation.

A l'exception de rw_data, les noms et les types de données de variables utilisées sont en correspondance avec ceux de la structure DRIS, utilisée avec des langages de type Texte.

Les type et taille de ces variables sont décrits dans le manuel DinkeyPRO, chapitre STRUCTURE DRIS. En voici un extrait :

Champ	Taille	Type	Entrée/ Sortie	Description
header	4	char	Entrée	Doit être initialisée avec la valeur 'DRIS'.
size	4	int	Entrée	Taille de la structure DRIS.
seed1	4	int	Entrée	Clé d'encodage aléatoire utilisée pour le cryptage DRIS et pour le cryptage des données.
seed2	4	int	Entrée	Autre clé d'encodage (Voir description ci-dessus).
function	4	int	Entrée	Permet, en méthode API, d'indiquer la fonction que vous souhaitez utiliser lors de l'appel de DDPProtCheck. Attention à n'utiliser qu'une seule fonction à la fois. Pour plus d'infos sur la liste des fonctions, voir la rubrique Fonctions
flags	4	int	Entrée	Le champ Flag vous permet d'indiquer des options pour chacune des fonctions. Vous pouvez utiliser une ou plusieurs valeurs en les associant avec OR. Pour la liste, consulter la section 'Flags'.
execs_dec	4	uint	Entrée	Décrémente le champ "exécutions" de la valeur indiquée en entrée. La valeur doit être positive.
data_crypt_key_num	4	int	Entrée	Indique quelle sera la clé de cryptage/décryptage utilisée pour le traitement des données sécurisées.
rw_offset	4	int	Entrée	Offset de lecture /écriture de la zone de données (positionnement dans la zone).
rw_length	4	int	Entrée	Longueur de la zone de données à lire ou à écrire.
rw_data_ptr	4/8	Pointer*	Entrée	Adresse de la zone de données à lire ou à écrire.

alt_prog_name	256	Char	Entrée	Permet d'indiquer à la fonction DDProtcheck en méthode API un nom de programme appelant différent de celui du programme réellement utilisé pour l'appel de cette fonction. Doit être terminé par une chaîne ASCII Null.
var_a	4	int	Entrée	Variable à passer à l'algorithme.
var_b	4	int	Entrée	Variable à passer à l'algorithme.
var_c	4	int	Entrée	Variable à passer à l'algorithme.
var_d	4	int	Entrée	Variable à passer à l'algorithme.
var_e	4	int	Entrée	Variable à passer à l'algorithme.
var_f	4	int	Entrée	Variable à passer à l'algorithme.
var_g	4	int	Entrée	Variable à passer à l'algorithme.
var_h	4	int	Entrée	Variable à passer à l'algorithme.
alg_number	4	int	Entrée	Indique le numéro de l'algorithme utilisateur à exécuter.
ret_code	4	int	Sortie	Code d'erreur.
ext_err	4	int	Sortie	Code d'erreur étendu.
type	4	int	Sortie	Type de Dongle Dinkey (1 = Pro, 2 = FD)
model	4	int	Sortie	Modèle de Dongle (1 = Lite, 2 = Plus, 3 = Net 1, 4 = Net 5, 5 = Net 10, 6 = Net 50, 7 = Net illimité).
sdsn	4	int	Sortie	Numéro de série développeur (SDSN).
prodcod	12	char	Sortie	Code produit (Terminé par Null).
dongle_number	4	uint	Sortie	Numéro de série du dongle.
update_number	4	int	Sortie	Prochain numéro de mise à jour (Démarre à 1).
data_area_size	4	uint	Sortie	Taille allouée à la zone de données sécurisées (en octets).
max_alg_num	4	int	Sortie	Plus grand numéro d'algorithme existant.
execs	4	int	Sortie	Exécutions restantes.
exp_day	4	int	Sortie	Jour d'expiration (nombre).
exp_month	4	int	Sortie	Mois d'expiration (nombre).
exp_year	4	int	Sortie	Année d'expiration (nombre).
features	4	uint	Sortie	Mot de fonction.
net_users	4	int	Sortie	Nombre maximum d'utilisateurs réseaux simultanés.
alg_answer	4	int	Sortie	Résultat de l'exécution de votre algorithme en fonction de l'algorithme utilisé et des variables d'entrée du DRIS dans var_a...var_h.
fd_capacity	4	uint	Sortie	Capacité en octets de la zone de données Flash Disk du dongle détecté. Seulement disponible pour les clés de type FD. La valeur est actuellement de 1Go mais pourra être complétée par des capacités supérieures dans le futur.
fd_drive	4	Char	Sortie	Unité locale attribuée à l'espace Flash Disk du dongle exemple : f:\

*4 octets pour les codes 32 bits et 8 octets pour les codes 64 bits

NB si la valeur "no limit" ou "illimité" est utilisée, la valeur mémorisée est -1.

rw_data est un tableau d'octets utilisé pour la transmission des données à lire/écrire dans la zone de données sécurisée de la clé. (Modèles Net et Plus uniquement).

4. Adaptation du code exemple à votre projet

Le code exemple est fourni prêt à l'emploi pour une utilisation avec nos clés de démonstration. Pour adapter ce code à votre propre SDK (si vous disposez d'un numéro de série développeur et de clés associées) il suffit de modifier les parties repérées par !!! dans le fichier apitest.c

* Modifier MY_SDSN il doit correspondre à votre numéro de série développeur (SDSN de demo = 10101)

* Modifier le code de MyAlgorithm (seulement si vous utilisez cette fonctionnalité)



- * Modifier le code de CryptDRIS (si vous cryptez le DRIS)
- * Modifier CryptApiData (si vous cryptez les données échangées entre votre EXE et dpwin32.dll)
- * Modifier le code de MyRWAlgorithm (Si vous cryptez les données envoyées à nos API en utilisant la fonction R/W algorithme)

Vous pourrez également personnaliser les messages d'erreur ainsi que la gestion des affichages de ces derniers.

5. Déploiement de votre application

Vous devrez fournir le fichier '*dpwin32.dll*' avec votre logiciel. Ce fichier doit être situé dans le même dossier que votre application. Si '*dpwin32.dll*' n'est pas trouvé, une erreur surviendra.



aplika

La Foltière - 37270 AZAY/CHER
Tél. 02 47 35 70 35 - Fax 02 47 35 70 25
e-mail : aplika@aplika.fr
www.aplika.fr