

Guide d'intégration



DINKEYPRO

Protection de logiciels en MATLAB 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 3.00 du 21/11/2011



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 MATLAB



1. Introduction	3
2. Utilisation de l'exemple dpwin32.dll (dpwin32test.m)	3
3. Utilisation de l'exemple dpMatlab.dll (dpmatlabtest.m)	4
4. Fonctions de vérification de la protection	4
4.1. Liste des fonctions GET	5
4.2. Valeur des fonctions du DRIS	5
4.3. Valeurs des flags du DRIS	6

Guide d'intégration DinkeyPRO/FD MATLAB

1. Introduction

Vous pouvez utiliser la méthode Shell et/ou la méthode API pour protéger vos applications Matlab.

Il existe 2 possibilités d'utilisation de la méthode API :

- Appeler `dpwin32.dll` (voir le code exemple du fichier `dpwin32test.m`)
- Appeler `dpMatlab.dll` (voir le code exemple du fichier `dpmatlabtest.m`)

La première méthode est recommandée si vous souhaitez exploiter les pleines fonctionnalités de la méthode de protection API. La seconde méthode est plus simple, avec une sécurité limitée et inclut une compatibilité avec les versions plus anciennes, ce que ne propose pas le code exemple `dpwin32.dll` (voir les remarques à la fin de ce fichier).

2. Utilisation de l'exemple `dpwin32.dll` (`dpwin32test.m`)

Il est délicat de définir la structure DRIS (structure des clés DinkeyPRO/FD) dans Matlab en raison du mélange de types *entiers* et *string* qu'elle contient. Dans ce code exemple, la structure DRIS est implémentée comme un tableau d'octets. Nous y avons également incluses divers fonctions *get* et *set* pour définir et retourner les différents éléments du DRIS.

Avant que vous ne puissiez utiliser le code exemple, vous devrez créer un fichier *m-file* des fonctions exportées par la dll. Pour cela, copiez les fichiers `dpwin32.h`, `dpwin32test.m` et `dpwin32.dll` dans un dossier que vous définirez comme dossier courant dans Matlab. Exécutez-alors la ligne de commande suivante :

```
Loadlibrary('dpwin32.dll','dpwin32.h','mfilename','dpwin32Functions.m')
```

Ceci créera le fichier `dpwin32Functions.m` référencé par le code source.

Le code exemple contient 10 fonctions. Vous n'avez pas à toutes les utiliser dans votre code. Utilisez simplement les fonctions adaptées à vos besoins et personnalisez votre propre intégration. Le code exemple représente un guide de mise en œuvre. Pour implémenter la protection d'une façon plus efficace, utilisez les techniques décrites dans le chapitre « Améliorer la protection » du manuel DinkeyPRO/FD.

Le code marqué par **!!!** indique les endroits où vous devez personnaliser le code exemple pour un fonctionnement correct :

- Modifiez la valeur de `MY_SDSN` pour la valeur de votre SDSN.
- Modifiez le code `MyAlgorithm` (si vous appelez l'algorithme utilisateur).
- Modifiez le code `CryptDRIS` (si vous cryptez le DRIS).
- Modifiez le code `MyRWAlgorithme` (si vous cryptez les données que vous envoyez à notre API en utilisant l'algorithme RW).

Il vous est possible, comme vous souhaiterez probablement, de remplacer nos messages d'erreur (par défaut) par vos propres messages.

Quelle(s) que soi(en)t la/les fonctions(s) que vous utilisez, vous devrez inclure toutes les routines communes ou « common routines » (c'est-à-dire toutes celles décrites ci-après, intégrant la fonction `init_dris`).

Remarque : Vous devez protéger `dpwin32.dll` et pas votre programme. Etant lié à la Dll, votre programme est protégé. La Dll protégée doit se trouver dans le même dossier que votre programme compilé pour fonctionner correctement.

Remarque : une liste des fonctions existantes et des valeurs de flags est disponible à la fin de ce document. Leur définition comme constantes globales serait préférable, toutefois, Matlab ne le permet pas.

3. Utilisation de l'exemple dpMatlab.dll (dpmatlabtest.m)

L'appel de dpwin32.dll est préférable car cette option ouvre à toutes les fonctionnalités et permet une sécurisation plus forte. Toutefois, pour des raisons de compatibilité avec les versions précédentes du SDK DinkyPRO/FD, nous incluons également les instructions d'appel à dpMatlab.dll.

dpMatlab.dll contient des fonctions simples pouvant être appelées à partir de Matlab. Ces fonctions utilisent toutes un DRIS (structure de données) intégré à dpMatlab.dll. Le principe consiste à appeler une des fonctions de vérification de la protection, puis d'appeler autant de fonctions Get... que vous souhaitez pour obtenir les paramètres contenu dans le DRIS. Ces valeurs demeurent valides jusqu'au test de la protection suivant leur mise à jour.

Remarque : le niveau de sécurisation général est limité par l'utilisation de ces fonctions simplifiées. Pour accroître la sécurité, vous pouvez :

- 1) Appeler dpwin32.dll au lieu de dpMatlab.dll.
- 2) Utiliser la méthode de protection Shell en complément (si vous déployez votre produit Matlab sous la forme d'un fichier exe).
- 3) Utiliser un algorithme (en utilisant la fonction DDSimpleCheckAlg).

Avant de pouvoir utiliser le code exemple, vous devez créer un fichier *m-file* des fonctions exportées par la dll. Pour cela, copiez les fichiers *dpMatlab.h*, *dpmatlabtest.m* et *dpMatlab.dll* dans un dossier que vous indiquerez comme dossier courant dans Matlab.

Saisissez ensuite la ligne de commande :

```
Loadlibrary('dpMatlab.dll','dpMatlab.h','mfilename','dpMatlabFunctions.m')
```

Ceci créera le fichier *dpMatlabFunctions.m* référencé par le code source.

Remarque : si vous souhaitez charger dpMatlab.dll en ligne de commande à partir de Matlab, utilisez le fichier dpMatlab.h à la place du fichier m-file précédemment créé :

```
Loadlibrary('dpMatlab.dll','dpMatlab.h','alias','Dongle')
```

4. Fonctions de vérification de la protection

int DDSimpleCheck(int flags)

Exécute un test de protection avec la valeur flags indiquée en paramètre et retourne un code d'erreur ou 0 si succès.

int DDSimpleCheckEx(int flags, unsigned int execs_decrement, char *alt_prog_name)

Exécute un test de protection avec la valeur flags indiquée et initialise la valeur execs_decrement du DRIS (Mettre Flags à 2 pour utiliser le décompte d'exécutions avec la valeur execs_decrement) et positionne la valeur alt_prog_name dans le DRIS (flags = 128)

Si flags n'est pas correctement positionné, execs_decrement et alt_prog_name sont ignorés.

Retourne un code d'erreur ou 0 si succès.

int DDSimpleCheckAlg(int flags, int alg_num, int var_a, int var_b, int var_c, int var_d, int var_e, int var_f, int var_g, int var_h)

Exécute un test de protection avec la valeur flags indiquée et permet également d'utiliser les algorithmes (valeur avec alg_num et variables a à h).

Retourne un code d'erreur ou 0 si succès.

Pour obtenir le résultat de l'algorithme, appelez la fonction GetAlgAnswer.

int DDSimpleCheckAlgEx(int flags, unsigned int execs_decrement, char *alt_prog_name, int alg_num, int var_a, int var_b, int var_c, int var_d, int var_e, int var_f, int var_g, int var_h)

Exécute un test de protection avec la valeur flags indiquée et permet également d'utiliser les algorithmes (valeur avec alg_num et variables a à h). Permet également d'utiliser execs_decrement et alt_prog_name

Retourne un code d'erreur ou 0 si succès.

Pour obtenir le résultat de l'algorithme, appelez la fonction GetAlgAnswer.

int DDWriteData(int flags, int rw_offset, int rw_length, unsigned char *data)

Exécute un test de protection avec la valeur flags indiquée et écrit rw_length octets de "data" à l'offset rw_offset dans la zone de données du dongle.

Retourne un code d'erreur ou 0 si succès.

int DDReadData(int flags, int rw_offset, int rw_length, unsigned char *data)

Exécute un test de protection avec la valeur flags indiquée et effectue la lecture de rw_length octets de "data" à partir l'offset rw_offset dans la zone de données du dongle. Les données seront lues à partir de l'argument "data".

Retourne un code d'erreur ou 0 si succès.

4.1. Liste des fonctions GET

Remarque : Toutes les fonctions ci-dessous retournent -2 si un test de protection n'a pas été effectué avant leur appel.

Si au moins un test de protection à déjà été effectué, la valeur de retour sera valide. Pour plus d'informations sur ces valeurs reportez-vous au manuel, rubrique 'Structure DRIS'.

int DDGetExtendedError(void)

int DDGetType(void);

int DDGetModel(void);

int DDGetSDSN(void);

unsigned int DDGetDongleNumber(void);

int DDGetUpdateNumber(void);

unsigned int DDGetDataAreaSize(void);

int DDGetMaxAlgNum(void);

int DDGetExecs(void);

int DDGetExpDay(void);

int DDGetExpMonth(void);

int DDGetExpYear(void);

unsigned int DDGetFeatures(void);

int DDGetNetUsers(void);

int DDGetAlgAnswer(void);

unsigned int DDGetFDCapacity(void);

int DDGetProdCode(char *prodcode); retour égal 0 si succès et remplit le Product Code (max 9 octets)

int DDGetFDDrive(char *fd_drive); retour égal 0 si succès et remplit notre fd_drive (max 128 octets)

4.2. Valeur des fonctions du DRIS

Spécifiez une seule fonction

Valeur	Nom dans le manuel	Description
1	PROTECTION_CHECK	vérifie la présence du dongle, vérifie les paramètres du programme...
2	EXECUTE_ALGORITHM	vérifie la protection et exécute l'algorithme indiqué avec les entrées spécifiées
3	WRITE_DATA_AREA	vérifie la protection et écrit dans la zone de données du dongle
4	READ_DATA_AREA	vérifie la protection et lit la zone de données du dongle
5	ENCRYPT_USER_DATA	vérifie la protection et crypte les données utilisateur
6	DECRYPT_USER_DATA	vérifie la protection et décrypte les données utilisateur
7	FAST_PRESENCE_CHECK	vérifie la présence d'une dongle correct avec la sécurité minimum, aucun flag autorisé

4.3. Valeurs des flags du DRIS

Dans la majeure partie des cas, la valeur du flag est 0. Si vous souhaitez utiliser plus d'un flag, ajoutez alors les valeurs.

Valeur	Nom dans le manuel	Description
1	DEC_ONE_EXEC	décrémente les exécutions de 1
2	DEC_MANY_EXECS	décrémente les exécutions du nombre spécifié dans execs_decrement
4	START_NET_USER	démarre un utilisateur réseau
8	STOP_NET_USER	arrête un utilisateur réseau (AUCUNE vérification de la protection n'est réalisée)
16	USE_FUNCTION_ARGUMENT	utilisez l'argument supplémentaire dans la fonction pour passer les données
32	CHECK_LOCAL_FIRST	recherche toujours un dongle local avant d'effectuer une recherche sur le réseau
64	CHECK_NETWORK_FIRST	recherche toujours un dongle réseau avant d'effectuer une recherche locale
128	USE_ALT_PROG_NAME	utilise le nom spécifié dans alt_prog_name plutôt que le nom du programme
256	DONT_SET_MAXDAYS_EXPIRY	si la date d'expiration n'a pas été calculée, ne pas le faire cette fois
512	MATCH_DONGLE_NUMBER	retrouve la recherche à dongle dont le numéro de série correspond à celui spécifié dans le DRIS



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