

dotNet Protector® Trace Decoder

Comment ça marche ?

TraceDecoder vous aide à gérer les erreurs d'exécution dans votre application.

Le but est d'obtenir la ligne de code source où l'erreur a lieu. On le fait généralement en chargeant la base de données de programme (informations de débogage, pdb) en même temps que l'assembly. Une fois protégé, les informations de débogage de l'assembly ne sont plus valides et donneraient de toute façon des informations sensibles.

Au lieu de déployer les pdb avec une application, dotNet Protector® peut construire au moment de l'exécution des traces décodables. Ces traces décodables ne peuvent être décodées qu'avec le(s) fichier(s) traceinfo correspondant(s) (les données contenues dans les traces décodables ne sont que celles disponibles à l'exécution, avec des noms maquillés, pas d'infos de fichiers sources).

Si vous collectez les traces décodables dans un module de rapport d'erreur, vous pouvez ensuite les décoder à l'aide de la bibliothèque TraceDecoder de dotNet Protector et de vos fichiers traceinfo.

Pour activer la génération des traceinfo, il vous suffit de cocher la case 'créer les informations de décodage de trace'.

Format des traces décodables

Une trace décodable est une chaîne de caractères multilignes (une ligne par trame décodable).

Une trame décodable a 2 sections séparées par le caractère NUL ('\0'). La première section est l'ID de traceinfo (en réalité le MVID du module) suivi d'informations de localisation de trame. La seconde section est la méthode obtenue par réflexion suivant le format :

[Nom_Module]Nom_Type_Complet::Nom_Méthode(Types_Paramètres)

Si vous voulez collecter les traces décodables, il est important que vous conserviez le formatage de la chaîne intact (un BinaryWriter fera l'affaire).

La Bibliothèque TraceDecoder (PvLogiciels.dotNetProtector.TraceDecoder.dll)

Bien sûr, TraceDecoder est protégé avec dotNet Protector, donc la runtime dotNet Protector est nécessaire pour l'exécuter.

La classe principale est la classe TraceDecoder PvLogiciels.dotNetProtector.TraceDecoder). Le constructeur instancie un TraceDecoder qui cherche par défaut les fichiers traceinfo dans le répertoire de données de dotNet Protector. En fait il enregistre un DirectoryInfoStore pour charger les traceinfo depuis leur emplacement par défaut. Vous pouvez implémenter vos propres classes dérivées de TraceInfoStore en implémentant la méthode GetModuleDebugInfoReader(Guid);

Cette méthode accepte l'ID de traceinfo comme paramètre et renvoie un BinaryReader sur les données de traceinfo (ou null si l'ID n'existe pas). Le stream sous-jacent est fermé par le TraceDecoder quand les données ont été lues.

Pour enregistrer votre TraceInfoStore personnalisé, appelez la méthode RegisterStore.

Pour décoder une trace, utilisez simplement la méthode DecodeTrace qui prend comme argument une trace décodable et retourne la version décodée.

La solution exemple TraceDecoderSample

FaultyApplication : une application bidon qui lève une exception chaque fois que l'on appuie sur un bouton, avec un gestionnaire de 'ThreadException' qui capte les exceptions, affiche la trace (la version maquillée donnée par le framework) et enregistre la trace décodable dans un fichier 'Errors.log'.

TraceReader : une application qui charge les fichiers log et affiche les traces décodées.

Protégez FaultyApplication en utilisant FaultyApplication.dpp et exécutez FaultyApplication. Lancez TraceReader pour décoder le journal.