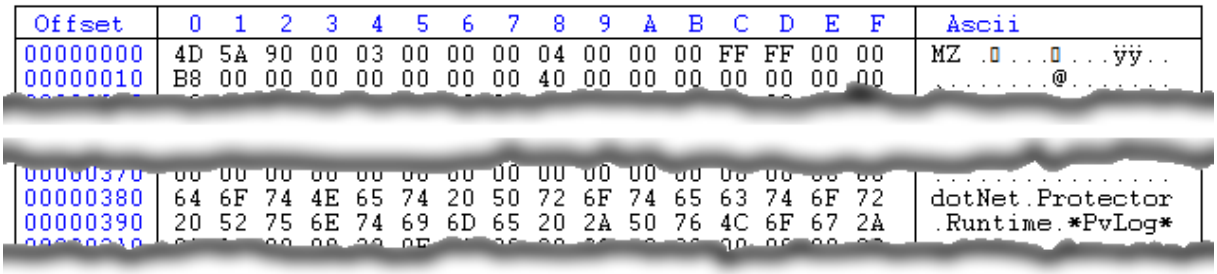
Structure de la signature PvLog

1. Signature pour la Runtime dotNet Protector

Un fichier image de Runtime de dotNet Protector est soit signé en utilisant Authenticode, soit signé en utilisant l'algorithme PvLog.
La signature Authenticode est en dehors de la portée de ce document.

Images runtime signées PvLog

Offset Fichier	Type	Description
0x380	BYTE[]	marquage (dotNet Protector Runtime *PvLog*)
0x3A0	DWORD	Offset du message signé exprimé depuis le début du fichier
0x3A4	DWORD	Taille du message signé



Procédé de géneration du hachage pour la runtime signée PvLog
Calculer le hachage PvLog d'une image de runtime est similaire au calcul de hachage Authenticode, sauf que le marquage et les pointeurs (de l'offset 0x380 à 0x3A7) doivent êtres omis.
Toutes les données dans les sections specifiés dans la table des sections doivent être hachées à l'exception des intervalles suivants:

- **Le champ CheckSum des champs spécifiques Windows du 'optional header'**. Cette check somme inclut tous les octets du fichier (incluant tout attribut de certificat dans le fichier). La check somme sera probablement différente apes avoir inséré la signature.
- **Le marquage de la Runtime et les pointeurs de message (offset 0x380 à 0x3A7)**
- **Informations concernant les certificats**. Les parties du ficher relatives à la signature ne sont pas incluses dans le calcul du hachage:

  Le champ 'Certificate Table' des directories du 'optional header'.

  La table de certificats et les certificats correspondants signalés par le champ ci-dessus.

  Pour calculer le hash, PvLogSigVerif classe les sections par adresse, et ensuite hache les séquences d'octets, omettant les intervalles exclus.

  - **Informations après la dernière section**. Tout ce qui est au-delà de la dernière section est omis. Cette zone contient généralement des informations de débogage et ne porte pas atteinte à l'intégrité proprement dite du programme exécutable. Il est littéralement possible de supprimer des informations de débogage à partir d'une image après un produit a été livré et ne pas affecter la fonctionnalité du programme. En fait, ceci est parfois fait comme une mesure d'économie de place. Il est intéressant de noter que les informations de débogage contenues dans les sections spécifiées de l'image PE ne peuvent pas être enlevées sans altérer la signature PvLog.

2. Signature pour les assemblies protégés par dotNet Protector

Comme la runtime non-authenticode, les assemblies protégés ont un marquage à l'offset 0x380.

Ce marquage inclut soit un identifiant d'éditeur de 64-bits si dotNet Protector est activé, ou 0 (64-bit) si l'assembly est protégé par une version d'évaluation. Il contient aussi un hash ordinateur de 64-bits (hash de l'ordinateur sur lequel l'assembly a été protégé).

Dans le cas d'une version d'évaluation, altérer le marquage ou la signature aura pour effet d'empêcher le programme de s'initialiser et donc de fonctionner.

L'image protégée est hachée en utilisant l'algorithme de hachage de nom fort et ce hachage est signé avec une paire de clés dont la clé publique est signée et horodatée par PV Logiciels et intégrée dans l'assembly protégé.

Offset Fichier	Type	Description
0x380	BYTE[]	Marquage (dotNet Protector)
0x390	QWORD	Hachage Editeur
0x398	QWORD	Hachage Ordinateur
0x3A0	BYTE[256]	Hachage signé (2048-bit)
0x4A0	BYTE[]	Clé publique signee et horodatée

Procédé d'authentification d'un assembly protégé

Verifier la signature du message et l'horodatage

Le message doit être signé par PV Logiciels / LA TESSOUALLE / FRANCE

Le certificate de signataire doit avoir été émis par une autorité de confiance (actuellement Symantec class 3 SHA256 Code Signing CA)

Le message doit inclure un horotadage RFC3161 (`1.2.840.113549.1.9.16.1.4`) contresigné par une autorité de confiance (actuellement COMODO)

Extraire la clé publique (ce qui a été signé)

Vérifier que le hachage signé correspond en calculant un hachage de nom fort sur le fichier et en vérifiant la signature.