Recherche de paramètres perdus d'un programme.

- Les articles -

Date de mise en ligne : vendredi 17 septembre 2004

Date de parution : 21 octobre 2003

Description :

Retrouver les paramètres (perdus) d'un programme, mais si c'est possible.

Environnement iSeries

Retrouver les paramètres (perdus) d'un programme, mais si c'est possible.

Problème : Comment appeler un programme dont on ne connaît pas les paramètres ? Personne ne les connaît ? Y'a pas de doc ? Le référentiel le boude ? Le CHEF est en vacances ? Plus les sources ? ....

Et pourtant on est OBLIGE de l'appeler. Y'a pas le choix. Peux pas faire autrement. Interdit !!!

Solution (un peu complexe, mais ça marche, si on est têtu) : si quelqu'un peut m'en proposer une autre, je suis preneur !

Pour simplifier, on parlera du programme PGMX

1) Dans un premier temps, on va récupérer le nombre de paramètres avec un classique

DSPPGM PGMX

Ce n'est pas indispensable mais ca pourra toujours servir pour vérifier la suite.

Admettons que l'on obtienne 7 : le pgm attend 7 paramètres !

(si vous obtenez 0, fermez ce document, la suite ne vous intéresse pas !)

2) Ensuite on récupère un dump de l'objet par la commande moins connue :

DMPOBJ PGMX

Attention des droits spéciaux peuvent être requis

On obtient un magnifique spool sur 2 colonnes, une en hexa, l'autre en texte « clair », du contenu de l'objet

On y retrouve toutes sortes d'infos comme les ordres GAP, des noms de variables, du texte inséré dans le corps de votre programme...

(On pourrait même en théorie construire un outil de reverse-ingenering avec ces infos)

Une recherche de l'instruction *ENTRY vous positionne sur du code abscons dans lequel on peut enfin apercevoir quelque chose qui ressemble à une liste de variables :

* ò x .IEF0012 .PLISTRT* * (Ñ Þ *ENTRY.1 êc ® *ENTRY.2* * @é PATREG.P $ áé PATTYP.P* * ãÑ ãé PATNUM.P åé PATVER.P* * " \é PATCHO.P +l ñé PATCDR.P* * ñß °é POPT.P íM .é *ENTRY â * * ¢ .PL001.1 îM ¥ .PL001.2 ë * * + .1000673 <µ ! .1000675 *l** *& .1000676 !C .1000276 * * ê .PL001 ô · .PL002.1* Ð ©* * .PL002.2 ä, í .1000702 î* * .PL002 ä] .PL003.1 ñÑ ¼ * *.PL003.2 " ß .PL003 , ½ .P*

Pour faciliter la lecture, je vous ai surligné en rouge les variables paramètres (si si, je peux le faire, ça ne mange pas de pain et ça évite au lecteur pressé de lire du texte qui ne sert à rien d'autre qu'à polluer le présent article et à faire grossir la taille de la page HTML d'où des temps de chargement plus lents, des dents qui... STOP, je m'égare...)

En théorie on pourrait décrypter les caractères qui suivent pour en déduire la longueur de chacun d'eux (calcul de l'offset en hexadécimal, et par écart trouver la taille ...). TROP DE BOULOT, et risque d'erreurs.

Heureusement il y a plus simple :

3) Pour récupérer la longueur des variables dont on connaît désormais le nom (le lecteur attentif a appris qu'elles sont en rouge quelques lignes plus haut) on va faire planter le programme (eh oui, exprès)

CALL PGMX

Un joli message système vous indique alors :

(C G S D F) PGMX 333 Utilisation d'un paramètre non transmis.

Il ne vous reste plus qu'a lui répondre **'D'** et à consulter le spool du dump généré en y recherchant vos variables :

```
PATREG   000646  CHAR(2)       '  '
 PATREG.P 000000  POINTER(SPP)
 SPACE OFFSET  1606                          '00000646'X
 OBJECT        QCAWAREA

 PATTYP   000667  CHAR(1)       ' '
 PATTYP.P 000010  POINTER(SPP)
 SPACE OFFSET  1639                          '00000667'X
 OBJECT        QCAWAREA


 PATVER   000690  PACKED(2,0)   '0000'X
 PATVER.P 000030  POINTER(SPP)
 SPACE OFFSET  1680                          '00000690'X
 OBJECT        QCAWAREA
```

Post-scriptum :

C'est tout, facile non ? Merci de m'avoir lu jusqu'ici car c'est enfin fini.