CRACK bunny bricks v1.0

Une traduction française du tuto de flashtro avec une approche un peu différente
(Se reporter au crack de bad-dudes pour éviter les explications redondantes)

Quand on fait une copie du disque original on retrouve une erreur sur le cylindre 79, vraisemblablement une protection de type RNC. Lorsqu'on boot sur le disque copié, le chargement s'arrête après l'écran de présentation des auteurs, ce qui n'arrive pas, bien sûr avec le disque original

On rentre dans l'AR au moment où le track 79 est testé (le lecteur fait un bruit bizarre), on fait ***D*** et on se retrouve dans une sous routine (finie par RTS) aux alentours de $C09xxx...
($C09EA8 ici mais différent en fonction des configs mémoire de l'amiga et du moment où l'on rentre dans l'AR)

En désassemblant vers le haut, on voit que cette routine commence en $C09E82, on fait donc un ***FA C09E82*** qui nous donne $C09C9C, là encore une sous routine donc, de la même façon, en désassemblant vers le haut, on voit que cette sous routine commence en $C09C7C, on fait ***FA C09C7C***, et comme ça, en remontant à l'envers, on retrouve la toute première routine d'appel du test de track qui commence en $C09C5E.

```
^C09C5A ORI.B   #0,D0
^C09C5E BSR     00C09C7C
^C09C62 CMP.W   #1980,D0
^C09C66 BLS     00C09C76
^C09C6A CMP.W   #19E0,D0
^C09C6E BHI     00C09C76
^C09C72 CLR.W   D0
^C09C74 RTS
;-------------------------
```

Cette routine teste si le chiffre renvoyé en D0 après le test du track est compris entre 2 bornes ($1980 - $19E0). Si on fait le test avec le disque original, on obtient bien un chiffre dans cette fourchette, avec une copie, on obtient bien évidemment un chiffre en dehors. Si le test est bon, D0 est remis à zéro et on retourne à la routine d'appel
Ce n'est donc pas à proprement parler une clé qui est renvoyée au programme, mais juste un test de track (c'est ici que se situe la différence d'approche avec flashtro)

Pour trouver la routine d'appel, on fait ***FA C09C5E*** et on obtient $C116AA (chiffre différent si config mémoire différente)

```
;-------------------------------
^C116A0 MOVEM.L D0-D7/A0-A6,-(A7)
^C116A4 MOVEA.L 00C113A8,A0
^C116AA JSR     00C09C5E
^C116B0 TST.W   D0
^C116B2 BNE     00C116BC
^C116B6 MOVEM.L (A7)+,D0-D7/A0-A6
^C116BA RTS
;-------------------------------
```

Dans cette routine, on voit le test de D0, et un branchement ensuite. Il ne faut pas faire le branchement en $C116B2 pour que le jeu se lance puisque la routine précédente renvoie 0 si le disque est original.

Le problème, c'est qu'on ne peut pas simplement mettre des NOP à la place des instructions car, comme on est en adressage relatif (ici en $C116xx car l'amiga a 512 ko de mémoire supplémentaire, mais les valeurs seraient différentes dans d'autres configurations), les instructions vont être modifiées par le programme pour changer les adresses en $C116AA et $C116B2. La seule ligne qui ne changera pas est $C116B0.
On peut donc recourir à une astuce simple et changer le TST.W D0 en MOVEQ #0,D0, qui prend également 2 octets, et dont le résultat sera bien sûr toujours égal à 0 !

Il reste à transformer le code original sur la disquette. Il nous faut donc les opcodes des lignes $C116B0 et $C116B2 que l'on obtient grâce à la fonction ***M*** de l'AR (4A 40 66 00 00 08)
C'est une disquette DOS, le code se trouve dans le fichier T.X, donc on fait :
LM T.X, 30000 (l'AR nous renvoie $3F8EC comme borne de fin du fichier chargé)
F 4A 40 66 00 00 08 (l'AR nous renvoie $3BFBC)

On remplace le code et on sauve (cf. capture)

```
Ready.
lm T.X, 30000
Loading from 030000 to 03F8EC
Disk ok
f 4a 40 66 00 00 08
Search from: 000000 to: C80000
03BFBC
Ready.
d 3BFBC
^03BFBC TST.W   D0
a 3bfbc
^03BFBC moveq #0,d0
^03BFBE
sm T.X,30000 3F8EC
Disk ok
```

JEL 09/2018