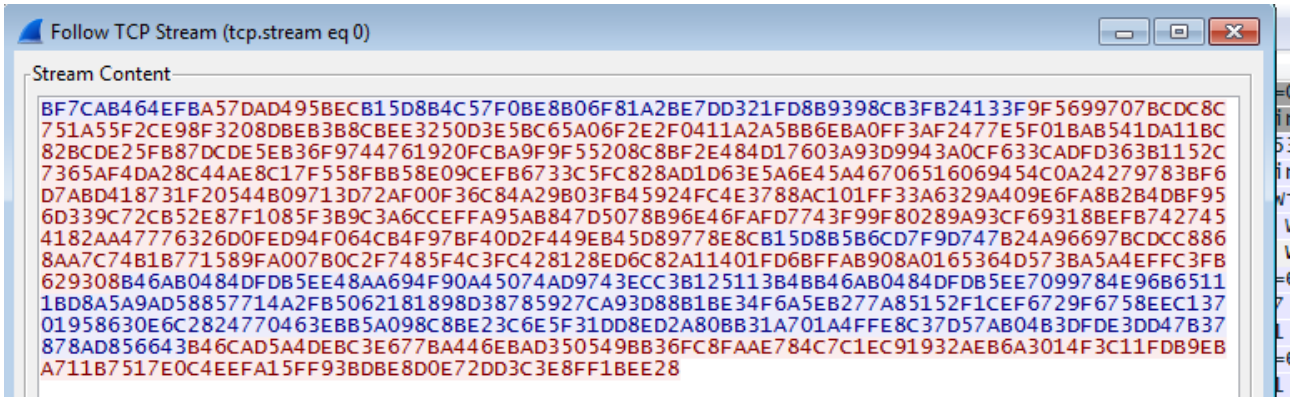
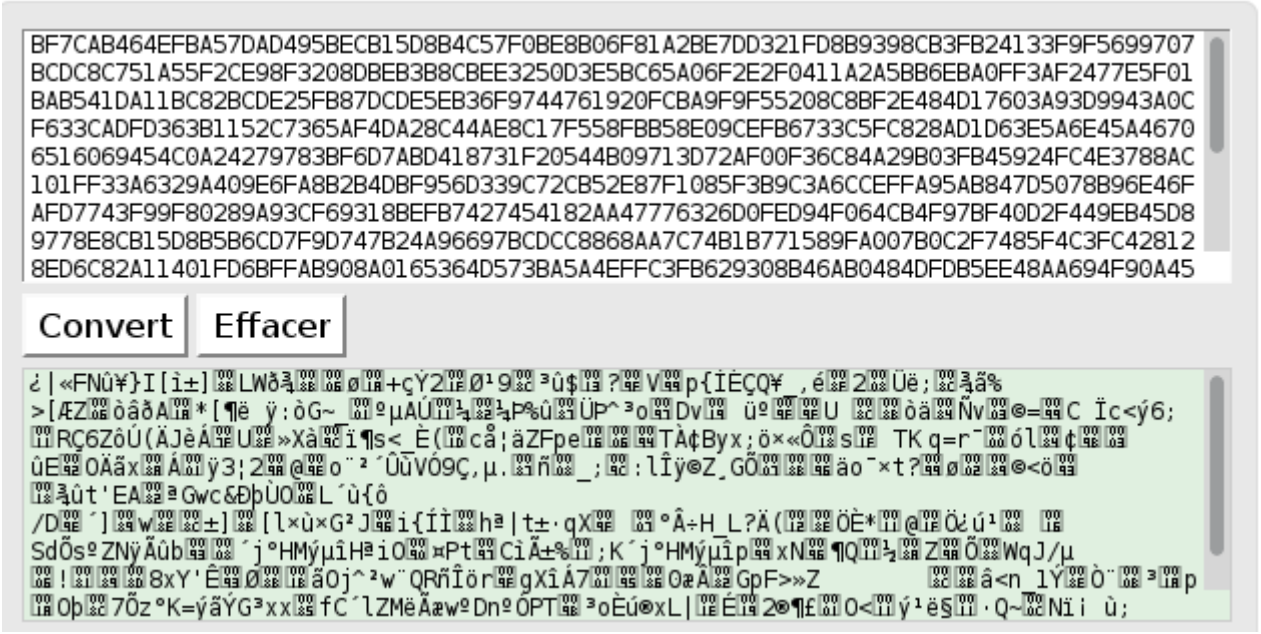
Première étape : Récupérer le clef de chiffrement

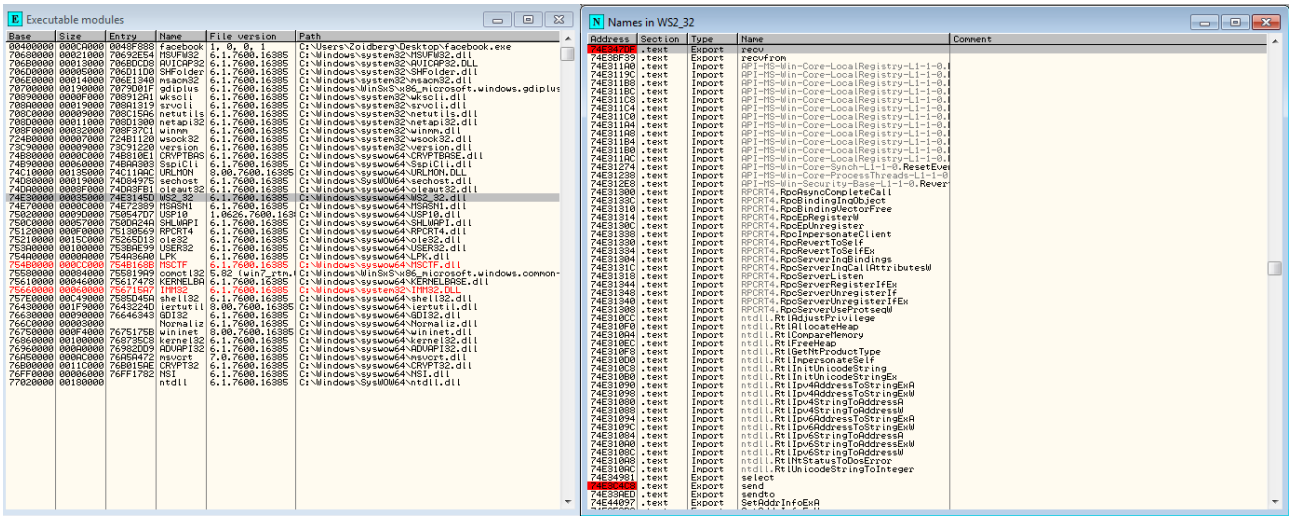
Un follow TCP stream sur le capture réseau nous donnes des données en hexadécimale sous leur représentation ASCII.



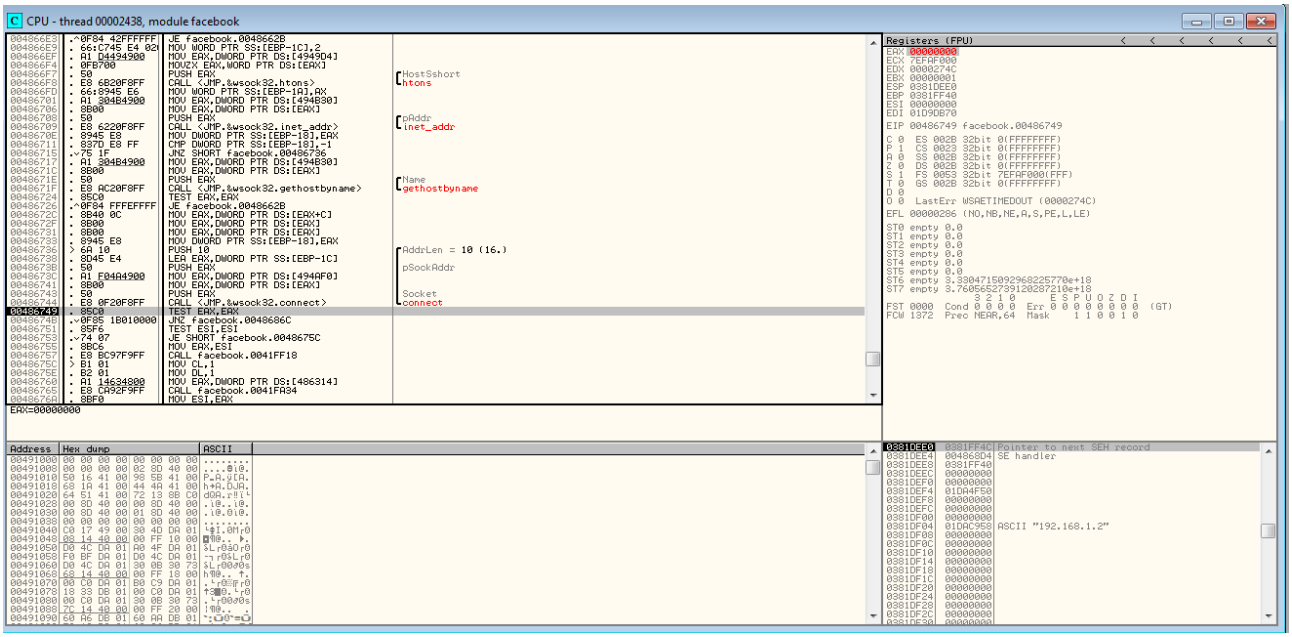
Une conversion hexadécimale vers ASCII ne produit aucune donnée intelligibles.



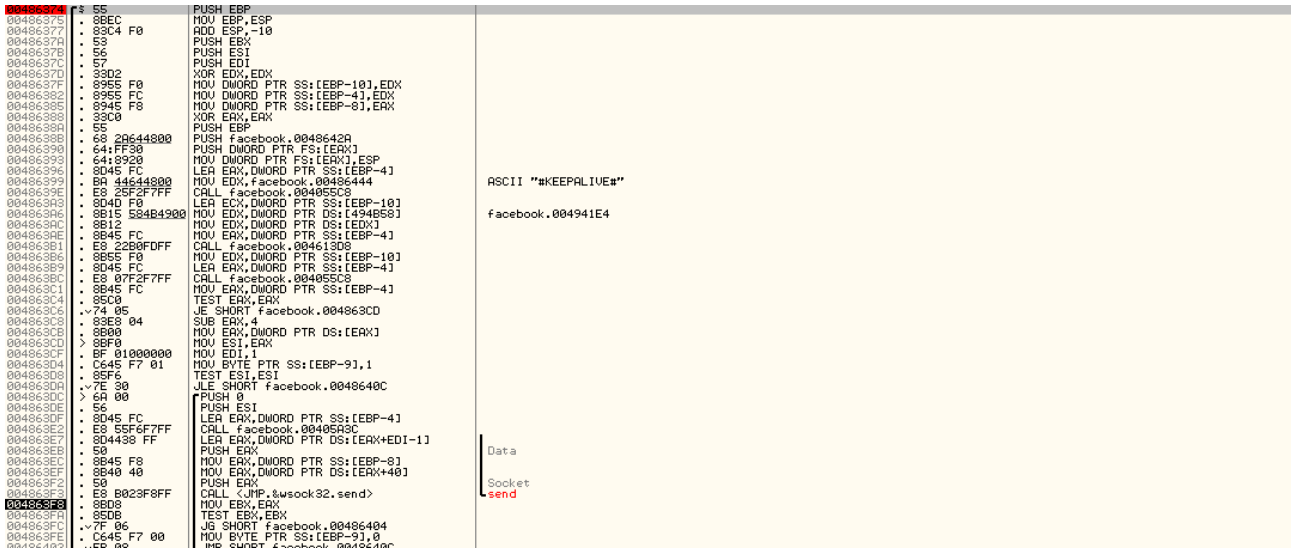
Les données sont donc probablement chiffrées par le malware. On va donc maintenant s'intéresser à cet exécutable pour retrouver l'algorithme et la clef utilisée. On va utiliser OllyDbg pour effectuer une analyse dynamique au sein d'une machine virtuelle. On va poser des breakpoints sur les fonctions send, recv et connect du module WS2_32.DLL, cela va nous permettre d'arriver au moment de l'envoi des packets et ainsi retrouver la fonction de chiffrement plus facilement.



On lance le programme et il s'arrête sur la fonction connect, le serveur de contrôle du malware n'existant pas on va tromper le programme en lui faisait croire qu'il a bien initialisé la connection. « Execute till return » pour attendre la fin puis on fais du pas à pas jusqu'à retrouver le module « facebook ». On arrive dans ce qui semble être la fonction qui intialise la connection avec le serveur. On voit que la valeur de retour est à « 0xFFFFFFFF » (dans le registre eax), on change celle-ci à 0 pour tromper le malware et on continue l'exécution.



Il faut procéder à deux fois cette manipulation avant que le programme se stop sur « send ». Une fois de retour dans le module « facebook » on remarque une chaine de caractère intéressante avant l'appel à « send »



Keepalive est utilisé en HTTP pour maintenir la connexion entre deux requêtes. On a pas vu de chaine « #KEEPALIVE #» dans la capture réseau, elle doit donc être chiffré dans la fonction qui a appelé « send ». On pose donc un breakpoint au début de celle-ci. On continue d'appliquer la technique pour tromper le programme sur « connect ». Et le malware finit par s'arrêter au début de la fonction. En procédant pas à pas on remarque une fonction qui prend en paramètre : « #KEEPALIVE# » et « #KCMDDC51#-890 » et retourne une chaine hexadécimale comme dans notre capture ! Cela doit être surement notre fonction de chiffrement.

On a maintenant la clef, il faut déterminer l'algorithme utilisé pour chiffrer les données. Le code initialise un tableau de 256 entrées grâce à la clef puis effectue pour chaque octet une série de modification avant de faire un XOR avec l'octet des données à chiffrer. Cette algorithme correspond au RC4. On peut s'aider du plugin HexRays d'IDA, ou, si on la personne à découvert le nom du malware (scan sur virustotal par exemple), retrouver l'algorithme – mais également la clef - utilisé sur les nombreuses analyses disponibles sur le web.

On peut maintenant déchiffrer les communications :

```
from Crypto.Cipher import ARC4

def decrypt(buf):
        buf=buf.decode("hex")
        cipher= ARC4.new("#KCMDDC51#-890")
        debuf=cipher.decrypt(buf)
        print debuf

decrypt("BF7CAB464EFB")
decrypt("A57DAD495BEC")
decrypt("B15D8B4C57F0BE8B06F81A2BE7DD321FD8B93E82B2FC27143A")
decrypt("9F5699707BCDC8C751A55F2CE98F3208DBEB3B8CBEE3250D3E5BC65A06F2E2F0
411A2A5BB6EBA0FF3AF2477E5F01BAB541DA11BC82BCDE25FB87DCDE5EB36F97447619
20FCBA9F9F522E8D8CF1E381D17603A93D9943A0CF633CADFD363B1152C7365AF4DA28C
44AE8C17F558FBB58E09CEFB6733C5FC828AD1D63E5A6E45A46706516069454C0A242797
83BF6D7ABD418731F20544B09713D72AF00F36C84A29B03FB45924FC4E3788AC101FF33A
6329A409E6FA8B2B4DBF956D339C72CB52E87F1085F3B9C3B6CCEFFA95AB847D5078B96
E46FAFD7743F99F80289A93CF69318BEFB7427454082AA47776326D0FED94F064CB4F97BF
4032F429AB45D89778E8C")
decrypt("D573BA5A4EFFC3FB629308")
decrypt("D573BA5A4EFFC3FB629308")
decrypt("A44D914F6CD1E2C24090675C98AC4770BB8E49F5CB88407C3E48DA4D028B9A812
46E4E39")
decrypt("D573BA5A4EFFC3FB629308")
decrypt("D57AB04B3DFDE3DD47B37878AD856643")
decrypt("B46CAD5A4DEBC3E677BA446EBAD350549BB36FC8FAAE784C7C1EC91932AEB6
A3014F3C11FDB9EBA711B7517E0C4EEFA15FF93BDBE8D0E72DD3C3E8FF1BEE28")
```

Et on obtient les données suivantes :

```
IDTYPE
SERVER
GetSIN192.168.1.1|4841375
infoesGuest16|192.168.1.1 / [192.168.1.1] : 1604|ZOIDBERG-PC / Zoidberg|4841375|0s|Windows
7 [7600] 64 bit ( C:\ )|- (Limited)||FR|Program Manager|{846ee340-7039-11de-9d20-806e6f6e6963-
1412106347}|39%|Fran?ais (France) FR /  -- |05/12/2014 at 13:59:40|5.3.0
#KEEPALIVE#
#KEEPALIVE#
RunPromptFLAG_DARKCOMET_1337_HACKERZ
#KEEPALIVE#
#BOT#CloseServer
BTRESULTClose Server|close command receive, bye bye...|Zoidberg
```

Le flag est : FLAG_DARKCOMET_1337_HACKERZ

Note : On peut également configurer un réseau local et émuler un serveur avec netcat, cela permet de ne pas avoir à tromper le programme et aller directement à l'étape du « send » . La fonction appelant « send » peut changer mais le principe reste significativement le même.