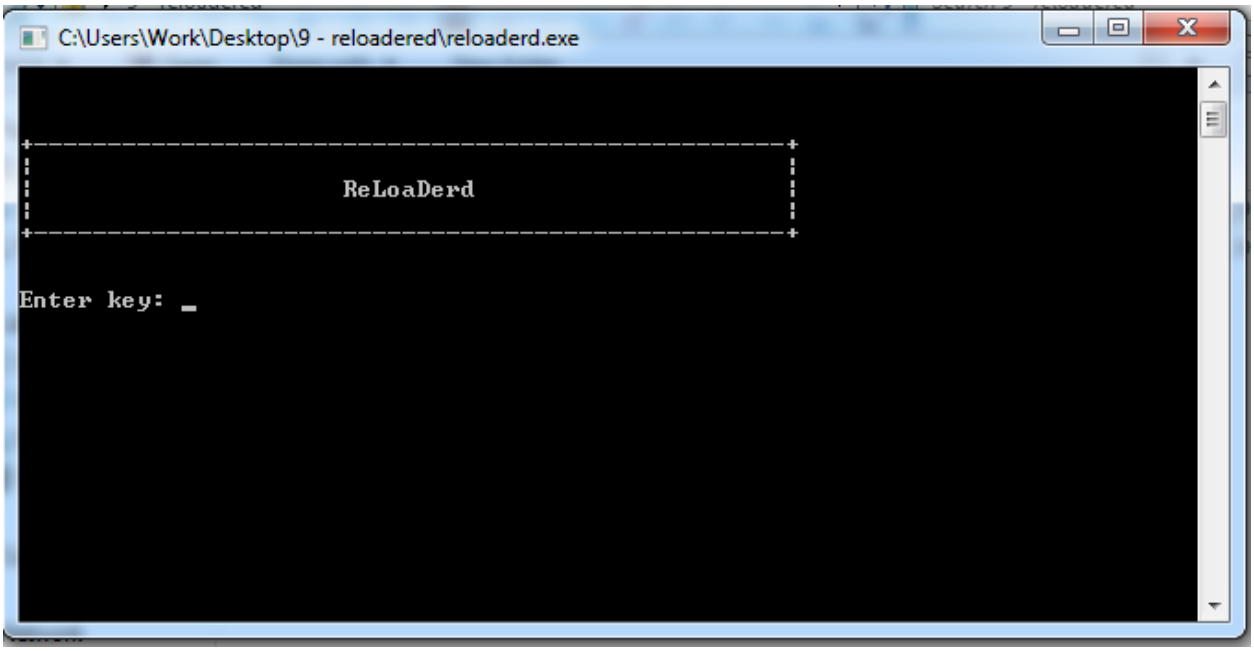
Reloadered

Chạy thử chương trình ta thấy:



Debug chương trình bằng x64dbg, search current module/string reference thì rất may mắn được danh sách khá ngắn gọn:

Ta thấy có 2 chuỗi Enter Key và mỗi chuỗi dẫn đến một đoạn code khác nhau, cho nên ta xem thử đoạn trang trí ReLoaDerd load ở đâu và bắt đầu từ đó

```
00011290    68 C0310100     push reloaderd.131C0
00011295    E8 06040000     call reloaderd.116A0
0001129A    68 FC310100     push reloaderd.131FC
0001129F    E8 FC030000     call reloaderd.116A0
000112A4    68 34320100     push reloaderd.13234
000112A9    E8 F2030000     call reloaderd.116A0
000112AE    68 FC310100     push reloaderd.131FC
000112B3    E8 E8030000     call reloaderd.116A0
000112B8    68 6C320100     push reloaderd.1326C
000112BD    E8 DE030000     call reloaderd.116A0
000112C2    83C4 14         add esp,14
000112C5    C3              ret
```
Breakpoint Not Set

Step out và ta có

```
00011680    E8 0BFCFFFF     call reloaderd.11290
00011685    E8 46FAFFFF     call reloaderd.110D0
0001168A    C3              ret
```

Vậy là trước đó sẽ có hàm call địa chỉ 0x11680, vào tiếp hàm 0x110D0

Ta step over hay run thì sẽ thấy break ở call 11060 và hiện ra prompt nhập Key

```
000110D9    A1 00400100     mov eax,dword ptr ds:[14000]
000110DE    33C5            xor eax,ebp
000110E0    8945 FC         mov dword ptr ss:[ebp-4],eax
000110E3    B9 21000000     mov ecx,21                          21:'!'
000110E8    E8 73FFFFFF     call reloaderd.11060
000110ED    8BC8            mov ecx,eax
000110EF    33C0            xor eax,eax
```

Và ngay bên dưới là đoạn so sánh để kiểm tra Key

```
000110F1    8A11            mov dl,byte ptr ds:[ecx]
000110F3    84D2            test dl,dl
000110F5    0F84 64010000   je reloaderd.1125F
00011100    0F1F4400 00     nop dword ptr ds:[eax+eax],eax
00011101    803C08 00       cmp byte ptr ds:[eax+ecx],0
00011105    75 F9           jne reloaderd.11100
00011107    83F8 0B         cmp eax,B                           B:'\v'
0001110A    0F85 4F010000   jne reloaderd.1125F
00011110    8A41 01         mov al,byte ptr ds:[ecx+1]
00011113    34 31           xor al,31
00011115    3C 5E           cmp al,5E                           5E:'^'
00011117    0F85 42010000   jne reloaderd.1125F
0001111D    0FBE41 02       movsx eax,byte ptr ds:[ecx+2]
00011121    25 FFFFFF0F     and eax,FFFFFFF
00011126    83F8 54         cmp eax,54                          54:'T'
00011129    0F85 30010000   jne reloaderd.1125F
0001112F    8079 0A 47      cmp byte ptr ds:[ecx+A],47          47:'G'
00011133    0F85 26010000   jne reloaderd.1125F
00011139    0FBE41 07       movsx eax,byte ptr ds:[ecx+7]
```
Breakpoint Not Set

Tiếp theo là đoạn XOR key với 1 đoạn dữ liệu trong memory địa chỉ 0x13108 dài 0x25 bytes, tuy nhiên byte thứ 25 được thế bằng \0 để kết thúc string

```
00011180    8BC7                mov eax,edi
00011182    33D2                xor edx,edx
00011184    F7F6                div esi
00011186    8A040A              mov al,byte ptr ds:[edx+ecx]
00011189    3287 08310100       xor al,byte ptr ds:[edi+13108]
0001118F    88843D FCFEFFFF     mov byte ptr ss:[ebp+edi-104],al
000111C6    8BC7                mov eax,edi
000111C8    47                  inc edi
000111C9    83FF 25             cmp edi,25                                  25
000111CC    72 E2               jb reloaderd.11180
000111CE    C68405 FCFEFFFF 00  mov byte ptr ss:[ebp+eax-104],0
```

```
1C 5C 22 00 00 17 02 62 07 00 06 0D 08 75 45 17  .\"....b.....uE.
17 3C 3D 1C 31 32 02 2F 12 72 39 0D 23 1E 28 29  .<=.12./.r9.#.()
69 31 00 39 00 00 00 00 45 6E 74 65 72 20 6B 65  i1.9....Enter ke
```

Kế tiếp là so sánh chuỗi sau khi giải mã với ".com" nếu không có xem như sai

```
00011201    C68405 FCFEFFFF 00  mov byte ptr ss:[ebp+eax-104],0
00011209    8D85 FCFEFFFF       lea eax,dword ptr ss:[ebp-104]
0001120F    68 74310100         push reloaderd.13174                        13174:".com"
00011214    50                  push eax
00011215    FF15 3C300100       call dword ptr ds:[<&strstr>]
```

Sau khi phân tích đoạn code check key ta tìm được key là RoT3rHeRinG

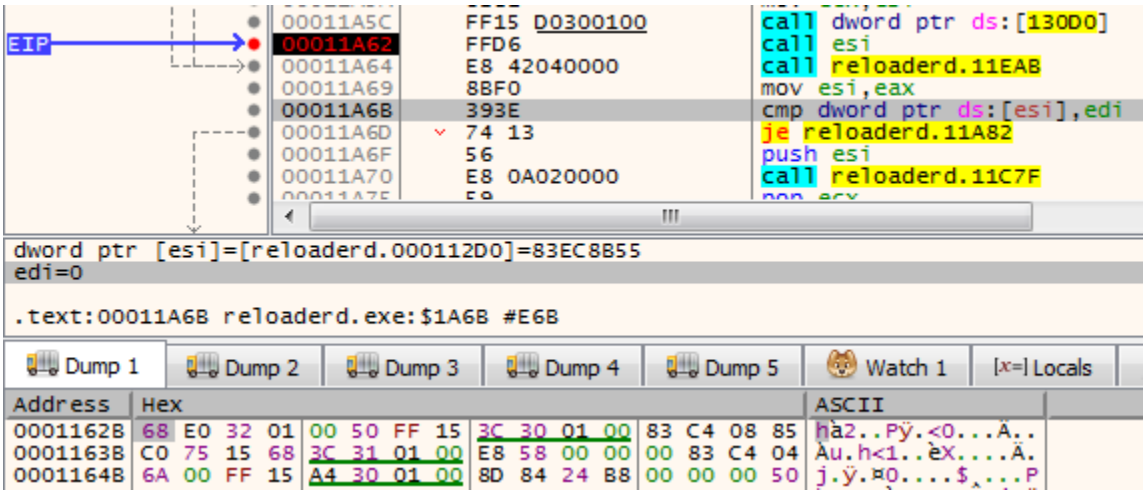
Gõ vào ta được:

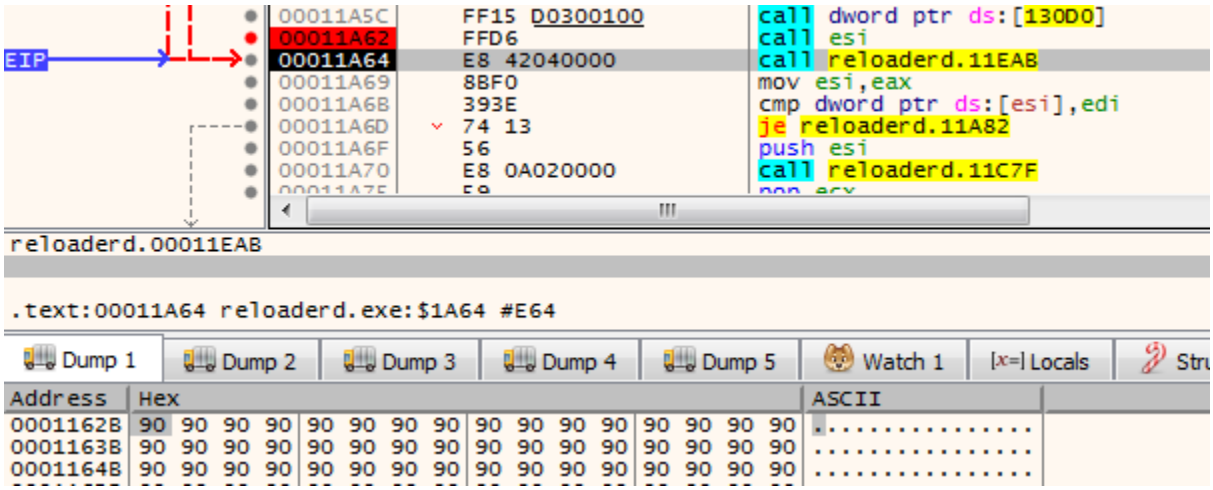
```
Enter key: RoT3rHeRinG
Here is your prize:

        N3v3r_g0nnA_g!ve_You_uP@FAKEFLAG.com
```

Vậy là đây là key giả, ta cần phải tìm key ở một chỗ khác, lần này ở string reference ta xem thử "@flare-on.com" nằm ở đâu, và ta thấy 1 hàm XOR khác xor giá trong trong stack với key và check với "@flare-on.com", và chuỗi này dài 0x34 bytes

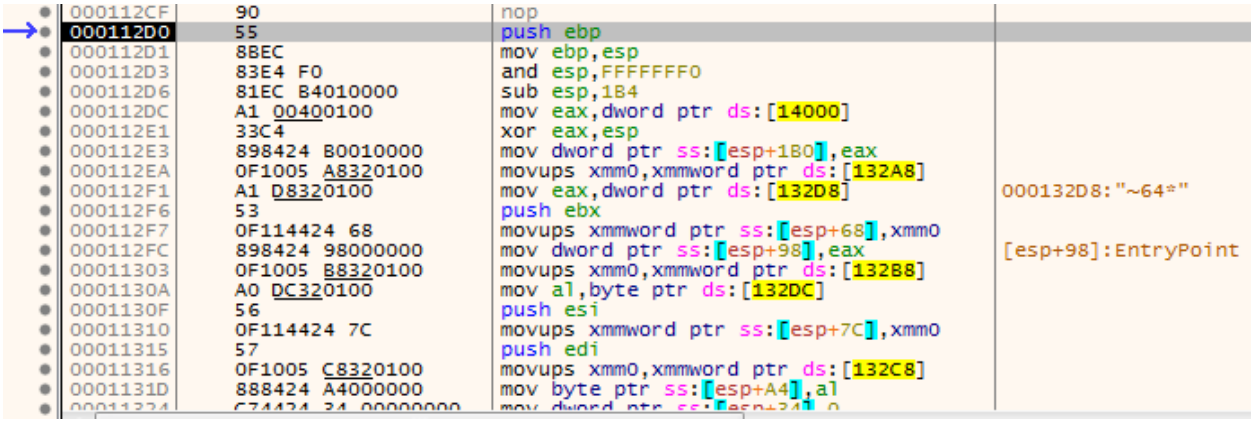
```
000115F0    8BC1                mov eax,ecx
000115F2    33D2                xor edx,edx                                 edx:EntryPoint
000115F4    F7F7                div edi
000115F6    8A0432              mov al,byte ptr ds:[edx+esi]                edx+esi*1:EntryPoint
000115F9    32440C 70           xor al,byte ptr ss:[esp+ecx+70]
000115FD    88840C B8000000     mov byte ptr ss:[esp+ecx+B8],al
00011604    8BC1                mov eax,ecx
00011606    41                  inc ecx
00011607    83F9 35             cmp ecx,35                                  35:'5'
0001160A    72 E4               jb reloaderd.115F0
0001160C    C68404 B8000000 00  mov byte ptr ss:[esp+eax+B8],0
00011614    40                  inc eax
00011615    3D 00010000         cmp eax,100
0001161A    73 54               jae reloaderd.11670
0001161C    C68404 B8000000 00  mov byte ptr ss:[esp+eax+B8],0
00011624    8D8424 B8000000     lea eax,dword ptr ss:[esp+B8]
0001162B    68 E0320100         push reloaderd.132E0                        132E0:"@flare-on.com"
```

Sau vài lần debug thì phát hiện mỗi khi chạy xong call esi thì sẽ xóa toàn bộ instruction đoạn code XOR và check với “@flare-on.com”

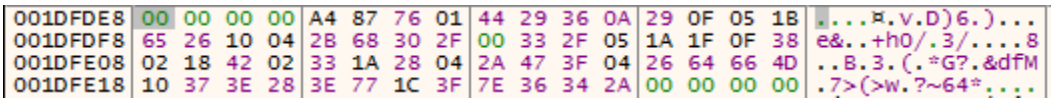




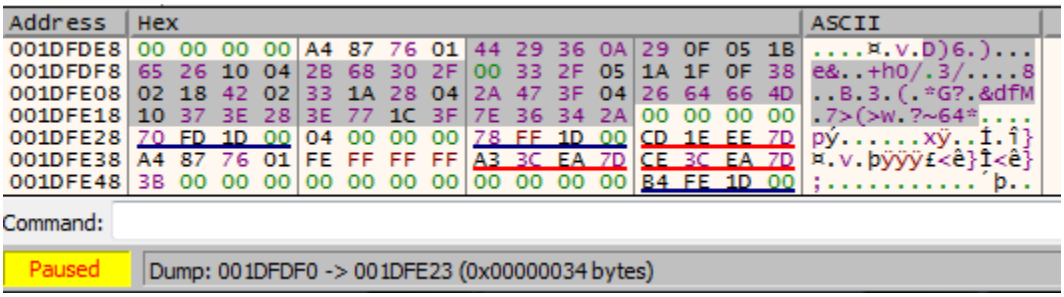
Check thử hàm gọi trong esi



Các instruction load dữ liệu từ 132A8 vào stack, sau khi load xong hết ta sẽ có stack là

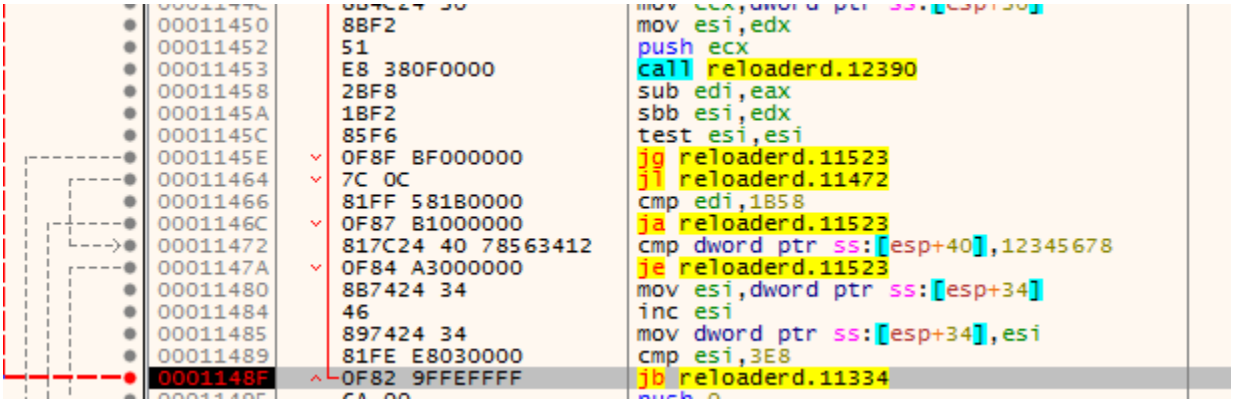


Ta đoán đây có thể là key đã được mã hóa, key dài 34 bytes, và dựa vào byte đầu tiên trong 132A8 ta có

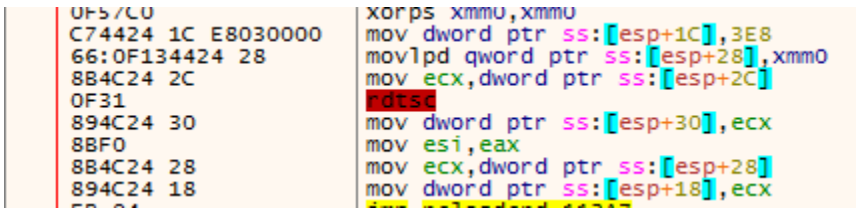


Lấy 13 bytes cuối cùng xor với "@flare-on.com", rất tiếc là không tìm được key, vậy có thể đây chưa phải là dữ liệu key mã hóa.

Xem tiếp các instruction kế tiếp ta thấy

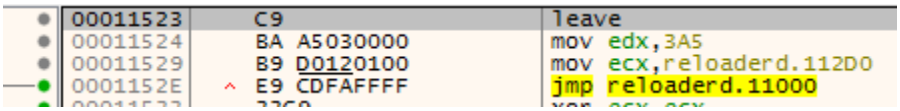


Khi debug chỗ này mỗi lần chạy mỗi lần chạy lại chương trình thì các giá trị được cmp lại khác nhau, kiểm tra lại vòng lặp ta thấy có các call đến rdtsc

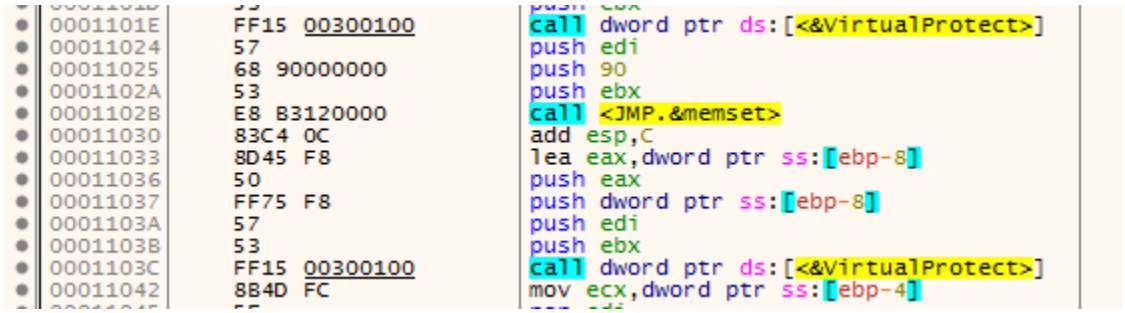


Hàm này sẽ lấy thời gian của máy bỏ vào edx:eax nên sẽ không có lần nào giống lần nào

Vòng lặp có số lần lặp là 0x3E8 để check và khi check địa chỉ 0x11523

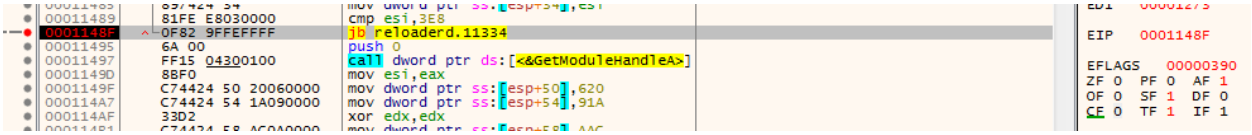


Check 0x11000



Hàm làm NOP dữ liệu của hàm 0x112D0

Ngoài ra khi check vòng lặp không thấy có sự thay đổi dữ liệu của stack, nên ta có thể break ở 0x1148F và chỉnh các flag để chương trình có thể thoái khỏi vòng lặp, ở đây chỉnh CF = 0



Tiếp tục ta lại gặp 1 vòng lặp khác

```
000114FA   3BC1              cmp eax,ecx
000114FC   75 10             jne reloaderd.1150E
000114FE   33C0              xor eax,eax
00011500   3B5484 50         cmp edx,dword ptr ss:[esp+eax*4+50
00011504   74 08             je reloaderd.1150E
00011506   40                inc eax
00011507   83F8 08           cmp eax,8
0001150A   72 F4             jb reloaderd.11500
0001150C   EB 15             jmp reloaderd.11523
0001150E   42                inc edx
0001150F   81FA CA130000     cmp edx,13CA
00011515   72 CA             jb reloaderd.114E1
00011517   64:A1 30000000    mov eax,dword ptr fs:[30]
0001151D   8078 02 00        cmp byte ptr ds:[eax+2],0
00011521   74 10             je reloaderd.11533
00011523   C9                leave
00011524   BA A5030000       mov edx,3A5
00011529   B9 D0120100       mov ecx,reloaderd.112D0
0001152E   E9 CDFAFFFF       jmp reloaderd.11000
00011533   33C9              xor ecx,ecx
```

Tương tự vòng lặp này cũng không thay đổi giá trị trong stack, ta lại chỉnh các flag lúc gặp các lệnh jump để ct chạy đến 0x11533 xor ecx, ecx

Tiếp theo là 1 vòng lặp và vòng lặp này có sửa giá trị trong stack và lặp 34 lần, ta xem trong dump stack đang chứa những giá trị gì

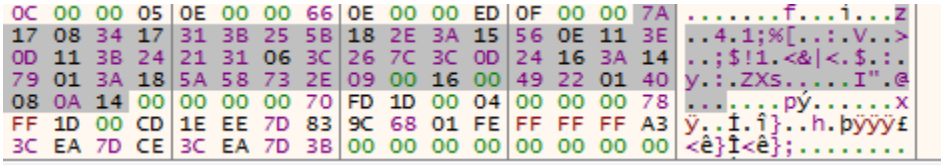
```
00011547   85FF              test edi,edi
00011549   74 21             je reloaderd.1156C
0001154B   B8 25499224       mov eax,24924925
00011550   F7E1              mul ecx
00011552   8BC1              mov eax,ecx
00011554   2BC2              sub eax,edx
00011556   D1E8              shr eax,1
00011558   03C2              add eax,edx
0001155A   C1E8 02           shr eax,2
0001155D   8D14C5 00000000   lea edx,dword ptr ds:[eax*8]
00011564   2BD0              sub edx,eax
00011566   8BC1              mov eax,ecx
00011568   2BC2              sub eax,edx
0001156A   75 04             jne reloaderd.11570
0001156C   304C34 70         xor byte ptr ss:[esp+esi+70],cl
00011570   46                inc esi
00011571   83FE 34           cmp esi,34
00011574   72 D1             jb reloaderd.11547
```

```
0C 00 00 05 | 0E 00 00 66 | 0E 00 00 ED | 0F 00 00 44 | .......f...i...D
29 36 0A 29 | 0F 05 1B 65 | 26 10 04 2B | 68 30 2F 00 | )6.)...e&..+h0/.
33 2F 05 1A | 1F 0F 38 02 | 18 42 02 33 | 1A 28 04 2A | 3/....8..B.3.(.*
47 3F 04 26 | 64 66 4D 10 | 37 3E 28 3E | 77 1C 3F 7E | G?.&dfM.7>(>w.?~
36 34 2A 00 | 00 00 00 70 | FD 1D 00 04 | 00 00 00 78 | 64*....pý......x
```

Chạy hết vòng lặp vài lần ta thấy đoạn dữ liệu trên không thay đổi và còn 1 vòng lặp lớn ở ngoài

```
00011571   83FE 34           cmp esi,34
00011574   72 D1             jb reloaderd.11547
00011576   41                inc ecx
00011577   81F9 39050000     cmp ecx,539
0001157D   72 B6             jb reloaderd.11535
0001157F   6A 01             push 1
00011581   6A 0E             push E
```

Chạy hết vòng lặp lớn ở ngoài ta có



Lúc này lấy 13 ký tự cuối cùng XOR với “@flare-on.com” ta có

```
['3', 'H', 'e', 'a', 'd', 'e', 'd', 'M', 'o', 'n', 'k', 'e', 'y']
```

Do đoạn code ở dưới check độ dài của key là < D vậy key này có 13 ký tự, vừa đúng chiều dài đoạn key vừa tìm được

XOR key với cả đoạn trên ta có được FLAG

I_mUsT_h4vE_leFt_it_iN_mY_OthEr_p4nTs?!@flare-on.com