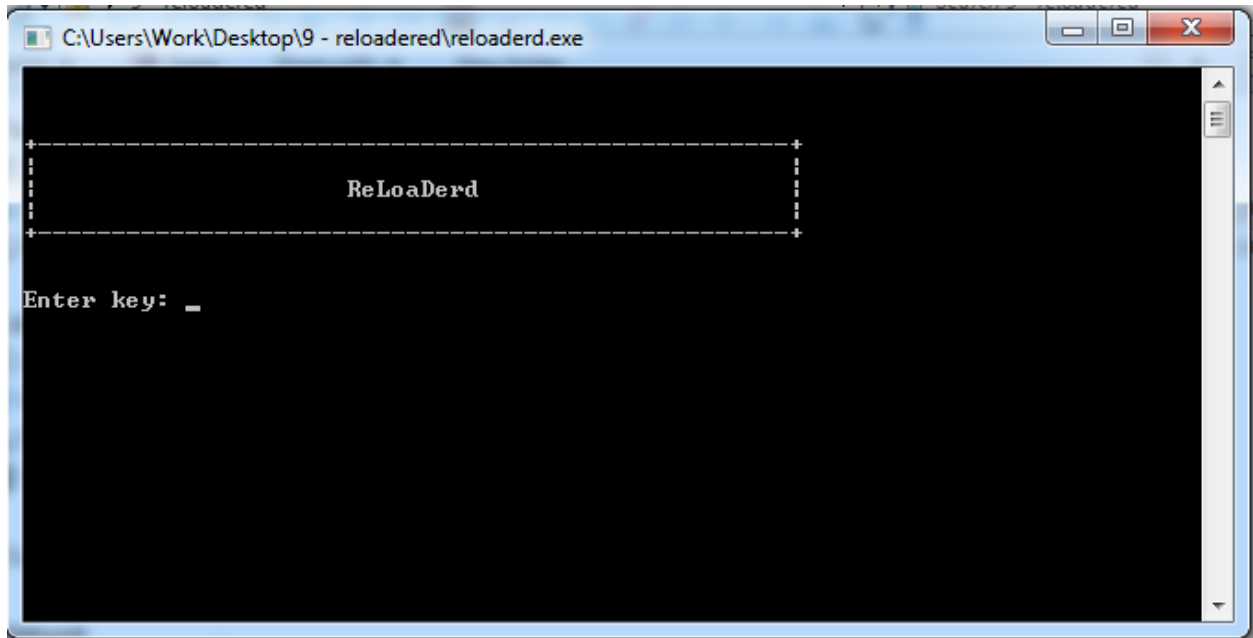
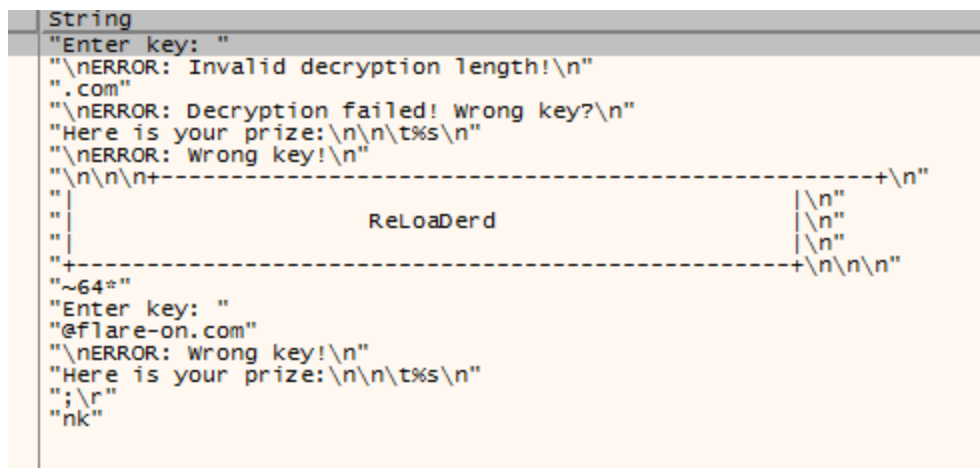


# Reloaded

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
000112B0	68 FC310100	push reloaderd.131FC
000112B8	E8 E8030000	call reloaderd.116A0
000112BD	68 6C320100	push reloaderd.1326C
000112C2	E8 DE030000	call reloaderd.116A0
000112C5	83C4 14	add esp,14
000112C5	C3	ret

Step out và ta có

00011680	E8 0BFCFFFF	call reloaderd.11290
00011685	E8 46FAFFFF	call reloaderd.11000
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	
000110F4	8A44	mov dl,byte ptr ds:[ecx]	

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	
00011101	0F1F4400 00	nop dword ptr ds:[eax+eax],eax	
00011105	40	inc eax	
00011107	803C08 00	cmp byte ptr ds:[eax+ecx],0	
0001110A	75 F9	jne reloaderd.11100	
0001110B	83F8 0B	cmp eax,8	B: '\v'
0001110D	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	0FB441 02	movsx eax,byte ptr ds:[ecx+2]	
00011121	25 FFFFFFFF	and eax,FFFFFFFF	
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	0FB441 07	movsx eax,byte ptr ds:[ecx+7]	

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]	
000111BF	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 68 65	i1.9...Enter ke
79 3A 20 00	0A 4E 52 52	4E 52 3A 20	52 73 6E 6E	...FPBP: When

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_uP0FAKEFLAG.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	edx:EntryPoint
000115F2	33D2	xor edx,edx	
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 88000000	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 88000000 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 88000000 00	mov byte ptr ss:[esp+eax+B8],0	
00011624	8D8424 88000000	lea eax,dword ptr ss:[esp+B8]	
00011628	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”

Assembly window showing instructions from 00011A5C to 00011A7C. The instruction at 00011A62 is highlighted in red. The memory dump shows a sequence of bytes: 68 E0 32 01, 00 50 FF 15, 3C 30 01 00, 83 C4 08 85, 00 00 00 50.

Assembly window showing instructions from 00011A5C to 00011A7C. The instruction at 00011A62 is highlighted in red. The memory dump shows a sequence of bytes: 90 90 90 90, 90 90 90 90, 90 90 90 90, 90 90 90 90, 90 90 90 90.

Check thử hàm gọi trong esi

Assembly window showing instructions from 000112CF to 00011324. The instruction at 000112D0 is highlighted in red. The assembly window shows instructions from 000112CF to 00011324.

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

001DFDE8	00 00 00 00	A4 87 76 01	44 29 36 0A	29 0F 05 1B	....P.v.D)6.)...
001DFDF8	65 26 10 04	2B 68 30 2F	00 33 2F 05	1A 1F 0F 38	e&...+h0/.3/....8
001DFE08	02 18 42 02	33 1A 28 04	2A 47 3F 04	26 64 66 4D	..B.3.(.*G?.&dfM
001DFE18	10 37 3E 28	3E 77 1C 3F	7E 36 34 2A	00 00 00 00	.7>(>w.?-64*....

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

Address	Hex	ASCII
001DFDE8	00 00 00 00 A4 87 76 01 44 29 36 0A 29 0F 05 1B	....P.v.D)6.)...
001DFDF8	65 26 10 04 2B 68 30 2F 00 33 2F 05 1A 1F 0F 38	e&...+h0/.3/....8
001DFE08	02 18 42 02 33 1A 28 04 2A 47 3F 04 26 64 66 4D	..B.3.(.*G?.&dfM
001DFE18	10 37 3E 28 3E 77 1C 3F 7E 36 34 2A 00 00 00 00	.7>(>w.?-64*....
001DFE28	70 FD 1D 00 04 00 00 00 78 FF 1D 00 CD 1E EE 7D	pý.....xy..i.i}
001DFE38	A4 87 76 01 FE FF FF FF A3 3C EA 7D CE 3C EA 7D	P.v.pýýýi<è}i<è}
001DFE48	38 00 00 00 00 00 00 00 00 00 00 00 84 FE 1D 00	.....p..

Command:

**Paused** Dump: 001DFDF0 -> 001DFE23 (0x00000034 bytes)

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

00011450	8BF2	mov esi,edx
00011452	51	push ecx
00011453	E8 380F0000	call reloaderd.12390
00011458	2BF8	sub edi,eax
0001145A	1BF2	sbb esi,edx
0001145C	85F6	test esi,esi
0001145E	0F8F BF000000	jg reloaderd.11523
00011464	7C 0C	jl reloaderd.11472
00011466	81FF 581B0000	cmp edi,1858
0001146C	0F87 B1000000	ja reloaderd.11523
00011472	817C24 40 78563412	cmp dword ptr ss:[esp+40],12345678
0001147A	0F84 A3000000	je reloaderd.11523
00011480	8B7424 34	mov esi,dword ptr ss:[esp+34]
00011484	46	inc esi
00011485	897424 34	mov dword ptr ss:[esp+34],esi
00011489	81FE E8030000	cmp esi,3E8
0001148F	0F82 9FEFFFFF	jb reloaderd.11334

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

0F57C0	xorps xmm0,xmm0
C74424 1C E8030000	mov dword ptr ss:[esp+1C],3E8
66:0F134424 28	movlpd qword ptr ss:[esp+28],xmm0
8B4C24 2C	mov ecx,dword ptr ss:[esp+2C]
0F31	rdtsc
894C24 30	mov dword ptr ss:[esp+30],ecx
8BF0	mov esi,eax
8B4C24 28	mov ecx,dword ptr ss:[esp+28]
894C24 18	mov dword ptr ss:[esp+18],ecx

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

00011523	C9	leave
00011524	BA A5030000	mov edx,3A5
00011529	B9 D0120100	mov ecx,reloaderd.112D0
0001152E	E9 CDAFFFFF	jmp reloaderd.11000

Check 0x11000

0001101E	FF15 00300100	call dword ptr ds:[<&VirtualProtect>]
00011024	57	push edi
00011025	68 90000000	push 90
0001102A	53	push ebx
0001102B	E8 B3120000	call <JMP.&memset>
00011030	83C4 0C	add esp,C
00011033	8D45 F8	lea eax,dword ptr ss:[ebp-8]
00011036	50	push eax
00011037	FF75 F8	push dword ptr ss:[ebp-8]
0001103A	57	push edi
0001103B	53	push ebx
0001103C	FF15 00300100	call dword ptr ds:[<&VirtualProtect>]
00011042	8B4D FC	mov ecx,dword ptr ss:[ebp-4]

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 thoát khỏi vòng lặp, ở đây chỉnh CF = 0

00011489	81FE E8030000	cmp esi,3E8	EDI 00001E79
0001148F	0F82 9FFFFFFFFF	jb reloaderd.11334	EIP 0001148F
00011495	6A 00	push 0	EFLAGS 00000390
00011497	FF15 04300100	call dword ptr ds:[<&GetModuleHandleA>]	ZF 0 PF 0 AF 1
0001149D	8BF0	mov esi,eax	OF 0 SF 1 DF 0
0001149F	C74424 50 20060000	mov dword ptr ss:[esp+50],620	CF 0 TF 1 IF 1
000114A7	C74424 54 1A090000	mov dword ptr ss:[esp+54],91A	
000114AF	33D2	xor edx,edx	
000114B1	C74424 58 AC000000	mov dword ptr ss:[esp+58],AAC	

Tiếp tục ta lại gặp 1 vòng lặp khác

000114FA	38C1	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 CDAFFFFF	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
00011548	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 18 65 26 10 04 28 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

00011570	46	inc esi
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

0C	00	00	05	0E	00	00	66	0E	00	00	ED	0F	00	00	7A	.....f...i...z
17	08	34	17	31	3B	25	58	18	2E	3A	15	56	0E	11	3E	..4.1;%[...V..>
0D	11	38	24	21	31	06	3C	26	7C	3C	0D	24	16	3A	14	..;\$!1.<& <.\$.:.
79	01	3A	18	5A	58	73	2E	09	00	16	00	49	22	01	40	y.:.ZXs.....I".@
08	0A	14	00	00	00	00	70	FD	1D	00	04	00	00	00	78	...pý.....x
FF	1D	00	CD	1E	EE	7D	83	9C	68	01	FE	FF	FF	FF	A3	ý..í.í}..h.pýýýf
3C	EA	7D	CE	3C	EA	7D	3B	00	00	00	00	00	00	00	00	<è}í<è};.....

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