mud lecture

[Rock lecture] 11. Put a trigger on the STR section

why do you ask
2014. 2. 18. 16:04

The MRGN section is 5100 bytes, so only 2 triggers can be inserted.
(It is possible to add a few more by overlapping triggers, but it is difficult)
Therefore, a trigger that is a little larger in size cannot be placed in this.
That's why we put the trigger after the STR section.

Today's goal:
1. Put a trigger on the STR section and execute the trigger.

We will run the loop we did in Lesson 9 in the STR section.

Putting a trigger on the STR section
Because we don't know where the trigger will be loaded (STR shorts go into dyna mically allocated memory)
You have to use a relocation table.

Let's look at the trigger in Lesson 9 again by applying only locoffset.

from trgformat import * import struct locoffset = 0x00000000 # offset of MRGN s ection # MRGN data triggers = [] # trigger A trigA = Trigger (players = [], conditi ons = [], actions = [SetMemory (0x0058D740 , SetTo , 0x00000000), SetMemor y (**locoffset + 8 + 320 + 32 * 0 + 16** , Add , 1), SetMemory (**locoffset + 8 + 320 + 32 * 0 + 20** , Add , 1), SetDeaths (Player1 , Add , 1 , 2), PreserveTrigger ()]) # trigger b B trigB = Trigger (players = [], conditions = [Deaths (Player1 , Exactly , 320 , 2)], actions = [SetMemory (**locoffset + 2408 + 4** , SetTo , 0xFF AE5D73), SetMemory (0x51A28C + 4 , SetTo , 0x51A28C), SetMemory (0x51A2 8C + 8 , SetTo , 0xFFAE5D73),]) trigAHeader = struct . pack ('<LL' , **locoffset + 2408** , **locoffset + 2408**) trigBHeader = struct . pack ('<LL' , **locoffset** , **loc offset**) data = b '' . join ([trigAHeader , bytes (trigA), trigBHeader , bytes (trig B)]) open ('payload.trg' , 'wb') . write (data)

whyask37's blog

+ 4: There are 4 bytes, next. **locoffset + 2408** (corresponding to trigger B)
+ 8 ~ 2407 : contents of trigger A. It corresponds to one trigger box.
+ 8~327 : condition of trigger A. All are filled with zeros.
+ 328~2375 : action table of trigger A
+ 328 ~ 359 : SetMemory(0x0058D740, SetTo, 0x00000000);
+360 ~ 391 : SetMemory(locoffset + 8 + 320 + 32 * 0 + 16, Add, 1); (Actually, it's probably the SetDeaths value.)
+ 376 : 4byte, **locoffset /4** + (8 + 320 + 32 * 0 + 16) / 4 - 0x 1628D9 value is stored. included in the PRT.
+ 380 : 4 bytes, there is 1, which is the number.
+ 384 : 2 bytes, unit number (0 here because EUP is used) is stored.
+ 386 : 1 byte, 0x2D corresponding to Set Deaths is stored.
+ 387 : 1 byte, 7 corresponding to Set To is stored.
+ 388 : 1 byte, 0x14 is stored here because it is Set Deaths . (Characteristic of Set Deaths.)
+ 389 ~ 391 : All zeros are filled. We're not going to deal with it.
+ 392 ~ 424 : SetMemory(locoffset + 8 + 320 + 32 * 0 + 20, Add, 1);
+ You can calculate the offsets as above. + 408 is included in the PRT.

If we calculate the PRT and ORT like this,
- 376, 408, and 2752 will be included in the PRT
- 0, 4, 2408, 2412 will be included in the ORT.

(0x 1628D9 = 0x 58A364 / 4. 58A364 starts death table)

The plan is as follows.
1. Assuming locoffset = 0, create a trigger and put it inside the STR section.
2. Later, **locoffset /4** + (8 + 320 + 32 * 0 + 16) / 4 - 0x 1628D9 can be added by adding the value of locoffset / 4 .
오프셋이 정해지지 않은 STR 섹션을 다뤄야 하니까 아마도 MRGN 트리거를 통해서 메모리를 조작해야겠군요.
3. 그리고 STR 섹션으로 PlayerTriggerStruct를 옮기면 됩니다.

정리하자면

whyask37's blog

```
from trigformat import * import struct locoffset = 0x00000000 # offset of MRGN section # STR 단락. 실제 동작. # trigger A trigA = Trigger ( players = [], conditions = [], actions = [ SetMemory ( 0x0058D740 , SetTo , 0x00000000 ), SetMemory ( locoffset + 8 + 320 + 32 * 0 + 16 , Add , 1 ), SetMemory ( locoffset + 8 + 320 + 32 * 0 + 20 , Add , 1 ), SetDeaths ( Player1 , Add , 1 , 2 ), PreserveTrigger () ] ) # trigger b B trigB = Trigger ( players = [], conditions = [ Deaths ( Player1 , Exactly , 320 , 2 ) ], actions = [ SetMemory ( locoffset + 2408 + 4 , SetTo , 0xFFAE5D73 ), SetMemory ( 0x51A28C + 4 , SetTo , 0x51A28C ), SetMemory ( 0x51A28C + 8 , SetTo , 0xFFAE5D73 ), ] ) trigAHeader = struct . pack ( '<LL' , locoffset + 2408 , locoffset + 2408 ) trigBHeader = struct . pack ( '<LL' , locoffset , locoffset ) data = b '' . join ([ trigAHeader , bytes ( trigA ), trigBHeader , bytes ( trigB )]) open ( 'str_payload.trg' , 'wb' ) . write ( data ) # MRGN trigger. STR 트리거를 초기화시키고 STR 단락으로 점프하는 역할을 한다. trigA = Trigger ( players = [], conditions = [], actions = [ # PRT initalization # 376, 408, 2752 is prt SetMemory (376 , Add , 0 ), SetMemory ( 408 , Add , 0 ), SetMemory ( 2752 , Add , 0 ), # ORT initalization # 0, 4, 2408, 2412 is ort SetMemory ( 0 , Add , 0 ), SetMemory ( 4 , Add , 0 ), SetMemory ( 2408 , Add , 0 ), SetMemory ( 2412 , Add , 0 ) ] ) trigAHeader = struct . pack ( '<LL' , 0 , 0 ) data = b '' . join ([ trigAHeader , bytes ( trigA )]) open ( 'mrgn_payload.trg' , 'wb' ) . write ( data ) # TRIG trigger stroffset = 2164 triggers = [] # MRGN 트리거를 초기화하는 로직이다. locoffset = 0x58DC60 # PRT, ORT에 따라서 MRGN 섹션에 값들 더하기 for i in range ( 31 , 1 , - 1 ): triggers . append ( Trigger ( players = [ Player1 ], conditions = [ Memory ( 0x5993D4 , AtLeast , 1 << i ), Deaths ( Player1 , Exactly , 0 , 1 ) ], actions = [ SetMemory ( 0x5993D4 , Subtract , 1 << i ), SetMemory ( locoffset + 4 , Add , 1 << i ), SetMemory ( locoffset + 8 + 320 + 32 * 0 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 1 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 2 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 3 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 4 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 5 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 6 + 16 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 0 + 20 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 1 + 20 , Add , 1 << ( i - 2 )), SetMemory ( locoffset + 8 + 320 + 32 * 2 + 20 , Add , 1 << ( i - 2 )), SetMe
```

et // 4), SetMemory (locoffset + 8 + 320 + 32 * 2 + 20 , Add , stroffset // 4), Set Memory (locoffset + 8 + 320 + 32 * 3 + 20 , Add , stroffset), SetMemory (locoff set + 8 + 320 + 32 * 4 + 20 , Add , stroffset), SetMemory (locoffset + 8 + 320 + 32 * 5 + 20 , Add , stroffset), SetMemory (locoffset + 8 + 320 + 32 * 6 + 20 , Add , stroffset)])) WriteTrg ('out.trg' , triggers)

코드를 짜서 버그가 안나면 그게 버그입니다. 디버깅을 30분정도 해서 위의 코드가 나왔습니다.
트리거 디버깅은 ollydbg를 가지고 하는게 좋습니다. ollydbg를 이용한 트리거 디버깅 방법은 다음 시간에 다루겠습니다.

이렇게 합니다. stroffset의 의미는 6강 참고. (원래 STR 단락의 내용이 2161byte기 때문에 ST R 단락의 트리거는 STR 단락의 2164byte 오프셋에 넣었습니다)

결과는 9강과 똑같게 0x0058D740 오프셋 부근을 반복문으로 채웁니다.
반복문이 STR 단락 위에 있는게 좋은거죠.

위에 코드가 제일 간단한 설명이긴 합니다만 그래도 글로 설명을 쓰는게 더 나을것같아서 첨언을 약간만 하자면

1. 이 부분

PRT initalization # 376, 408, 2752 is prt SetMemory (376 , Add , 0), SetMemor y (408 , Add , 0), SetMemory (2752 , Add , 0), # ORT initalization # 0, 4, 2408, 2412 is ort SetMemory (0 , Add , 0), SetMemory (4 , Add , 0), SetMemory (240 8 , Add , 0), SetMemory (2412 , Add , 0)

- locoffset + 8 + 320 + 32 * 0 + 16은 위의 85 D7 E9 FF에 해당하고,
- locoffset + 8 + 320 + 32 * 0 + 20은 위의 00 00 00 00에 해당하겠죠.

그래서 i가 31부터 2까지 다 돌면 MRGN의 액션은 다음과 같이 바뀔겁니다.
SetDeaths (addressSTR / 4 + 0xFFE9D785 , Add , addressSTR / 4, 0),

이것까지 합치면
SetMemory(locoffset + 8 + 320 + 32 * 0 + 16, Add, stroffset // 4),
SetMemory(locoffset + 8 + 320 + 32 * 0 + 20, Add, stroffset // 4),

이렇게 완성!
SetDeaths (stroffset / 4 + addressSTR / 4 + 0xFFE9D785 , Add , stroffset / 4 + addressSTR / 4 , 0),

addressSTR + stroffset이 STR 단락의 트리거의 오프셋이니까 위 액션은 다음과 같이 바꿔쓸수도 있죠.
SetDeaths (strtrgoffset / 4 + 0xFFE9D785 , Add , strtrgoffset / 4 , 0),

또는 이렇게.

whyask37's blog

why do you ask

This is whyask37's blog.

add neighbor

back to top

View in PC version