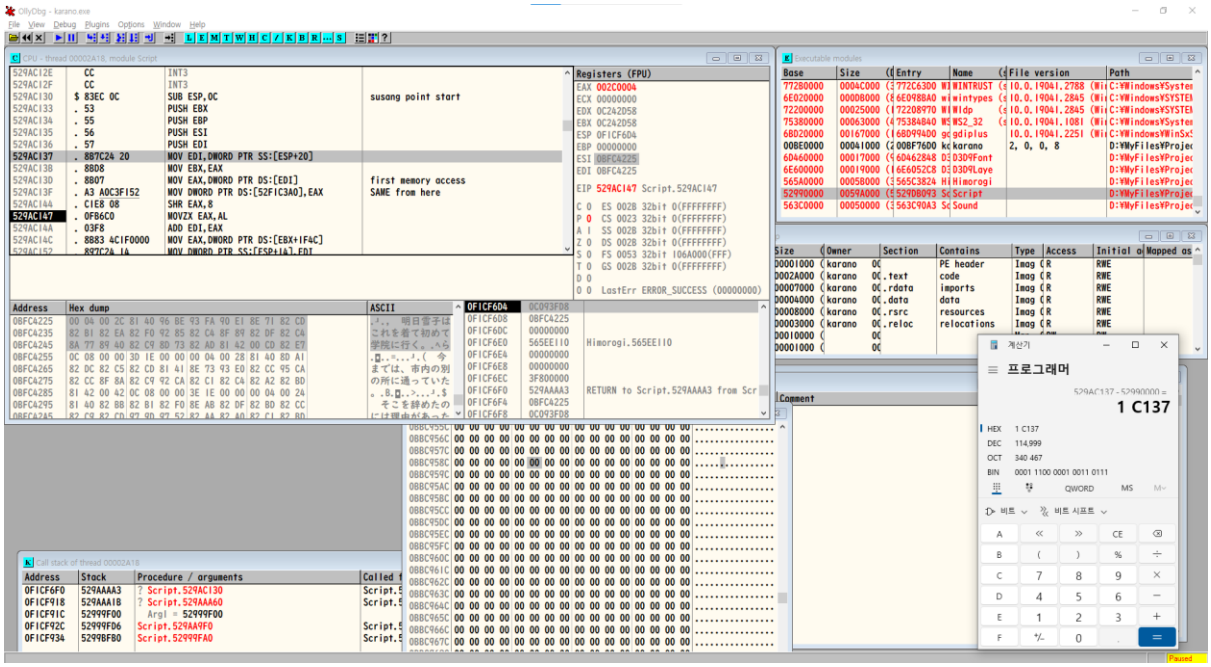


내가 언젠가 susang point start라고 코멘트를 남겨두었구나..

아무튼 수상한 지점이 시작되는곳인가본데, 이곳에 브포를 걸면 굉장히 깔끔하게 대사 한번 넘길 넘길 때 딱 한번씩 브레이크가 걸린다. 이전 코드파인딩 기록에서 여러 번 언급했던 부분인것같은데.. 일단 다시 주시해보자.

우선은 오늘 테스트해볼 첫번째 지점은 529AC137 – 52990000 = 0001C137 이고, [ESP+20]을 납치해볼것이다.

아래는 이것을 찾기까지의 과정이다.

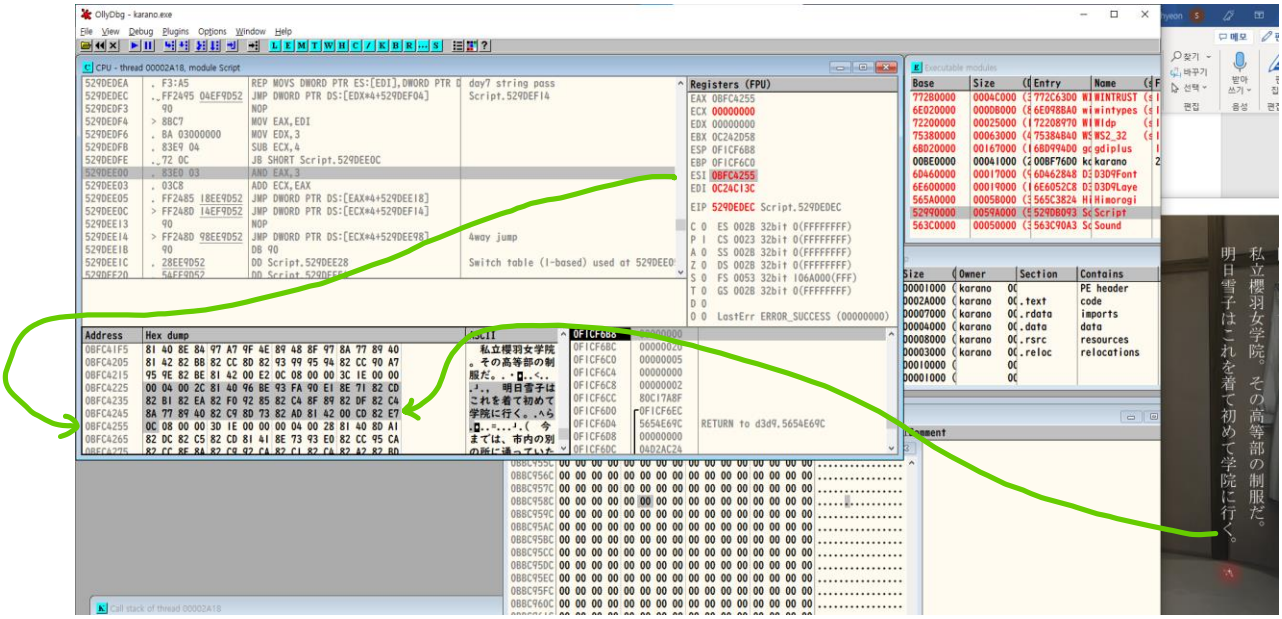
참고로 이번 테스트도 실패다.

굉장히 오랜만에 디버거를 켜봐서 뇌가 강제로 초기화된 상태로 레지스터를 보고있는데, 마침 눈에 띄는 부분이 있었다.

대사를 넘긴다 -> 브포에 걸린다. -> F9를 눌러 계속실행한다.

이 과정을 반복하면서, ESI가 계속 바뀌는 모습을 발견하였다.

ESI는 스크립트 원본이 들어있는 메모리를 가리키고있었다. 특히, F9를 눌러 계속 실행한 이후에 ESI는 방금 화면에 표시한 대사 메모리의 마지막 지점을 가리키고있었으니..



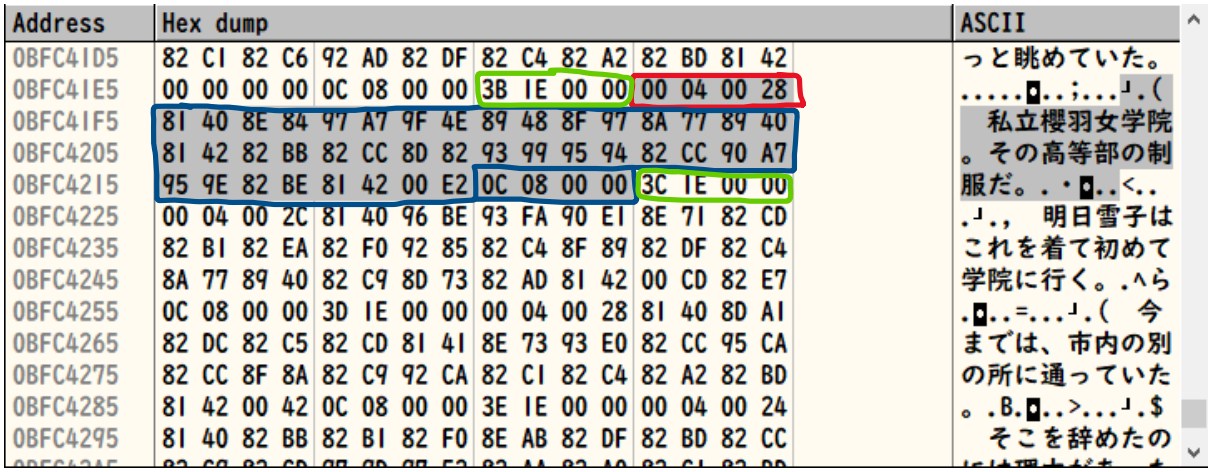
이렇게 ESI는 0BFC4255를 가리킨 상태로 종료된다는 것이다. 현재 Running상태인 게임화면에는 ～～～学園にいく。 라는 대사가 나와있고, 현재 ESI가 가리키고 있는 메모리는 그 대사의 마지막 지점 바로 이후이다.

눈길이 가지 않을 수가 없다.

이 이후에 메모리 브레이크를 걸고 ESI를 주시할것인데, 이를 수행할때의 디테일을 설명할것이다.

이것은 지금 뜯어보고있는 공허의소녀 리마스터에만 해당되는 구조일 테니 다른 게임과는 다를 가능성이 매우매우 높을것같다.

아래에서 원본 스크립트의 메모리 구조를 알아보도록 하자.



붉은 사각형으로 표시한 부분(00 04 00 28)이 "대사부분이며, 그 길이는 0x0028이다"라고 알려주는 것이다.

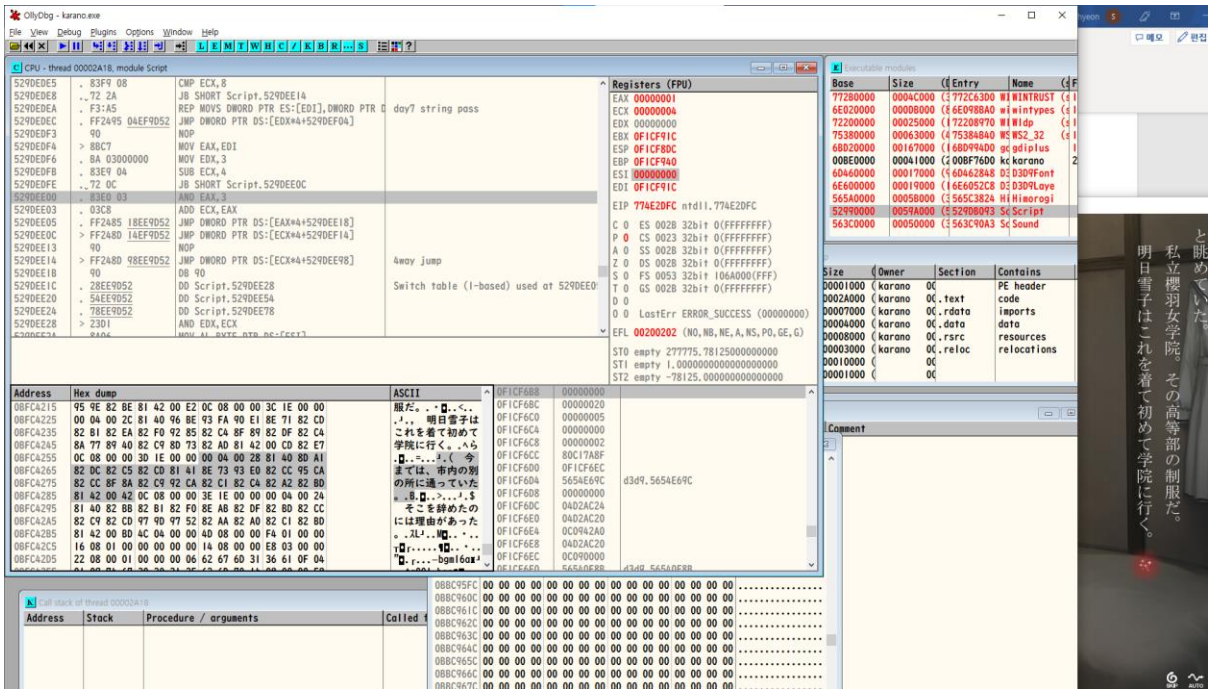
푸른 사각형으로 표시한 부분(0BFC41E5~)이 대사가 담겨있는 메모리이다.

푸른 사각형의 마지막 부분(0C 08 00 00)은 대사가 끝나는 지점을 알려주는 부분이다. 흔히 볼수 있는 줄바꿈표시, csv의 ₩t구분 등과 비슷하게 작동하는 표기인듯하다. 모든 대사의 맨 마지막부분에 0C080000이라는 값이 담겨있다. (읽힐때는 00 00 08 0C로 읽히는듯하더라)

초록 사각형은 어떤 대사에 붙어있는건지 확인하지 못했지만, 3B1E0000 -> 3C1E0000 -> 3D1E0000 처럼 계속 올라간다. 00001E3A 3B 3C...처럼 계속 증가하면서 현재 스크립트가 몇번째 대사인지를 알려주는 파트인듯하다. (이를 추측하는 근거로는 이전 코드파인딩 기록에서 찾아볼수 있는데, 코드를 살펴보다 보면 대사가 몇번째인지를 읽어오는 코드가 어딘가에 있다.)

학기중이라 시간이 별로 많지가 않다.

메모리 브레이크를 걸면 어떻게 되는지 알아보자.



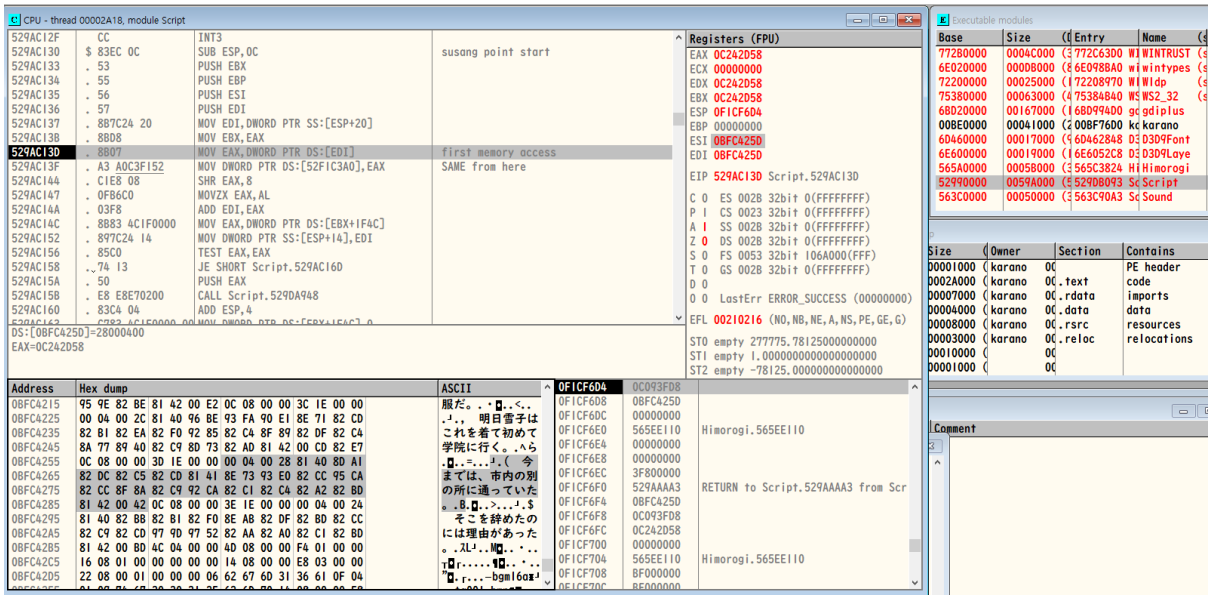
현재 Running상태이며 메모리 브레이크를 걸 지점은 위 사진의 회색 부분이다.

00 04 00 28은 "대사이고, 00 28 만큼의 길이"라고 알려주는 부분이므로, 다음대사의 시작지점쯤 될 것으로 추측가능하다.

메모리에 브포걸고 대사를 넘기면 게임화면은 변함이 없는채로 브레이크가 걸린다.

첫번째로 걸린 지점은 이전 코드파인딩 기록들에서 아주 제대로 구석구석 살펴본 부분인듯한데, 이번에 살펴볼 지점과는 좀 거리가 있어보이니 일단 넘겨보자.

그렇게 두번째로 걸린 지점이 바로 이번에 살펴볼 지점이 된다.



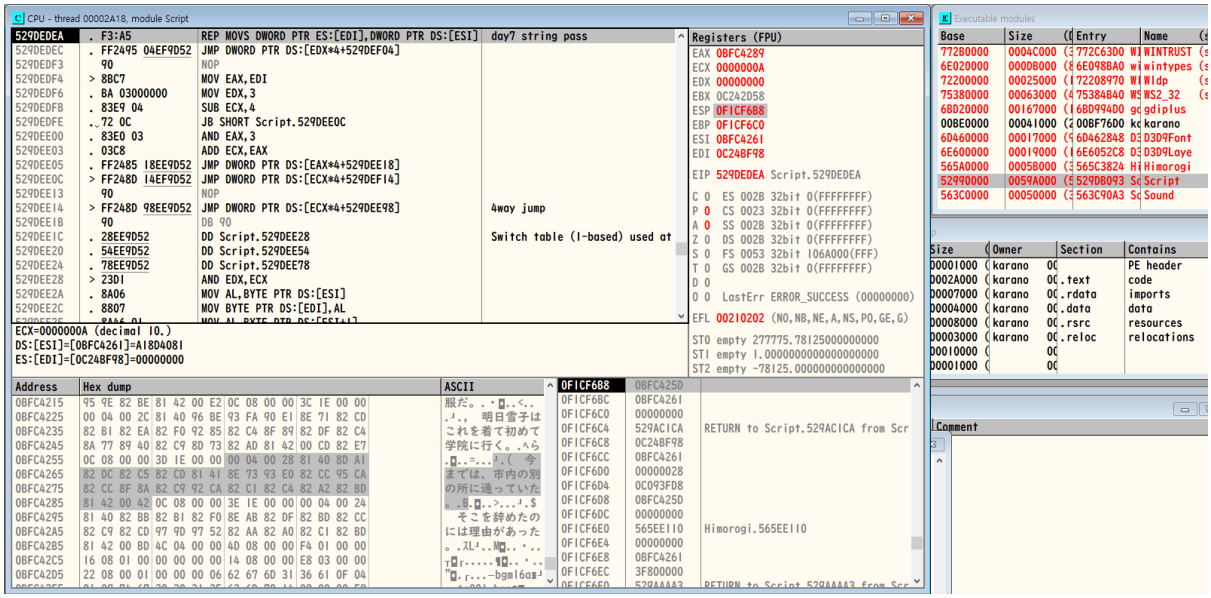
이때 브포에 걸린곳은 first memory access라고 comment를 남겨둔 부분이다.(사실 두번째긴하지만..)

레지스터를 보아하니 EDI가 ESI와 동일한 값을 갖고있다. 이게 어디서 온건가 보니 바로 두줄 위에서 EDI에 무언가 값을 저장하고있다.

529AC137 MOV EDI,DWORD PTR SS:[ESP+20]

스택에서 값을 가져와서 EDI에 넣는다. Susang point start이후에 PUSH가 4번뿐인데 스택포인터에서 0x20이나 올라가는걸 보니 susang point로 들어오기 이전에 언젠지는 모르지만 ESI를 스택에 넣는 부분이 있었나보다. (<-이번에 보진 않을거지만 잘 살펴봐야할지도..?)

일단 어디서 왔는지는 넘어가고 F9를 눌러 다음을 살펴보자.



뭔가 멀리 넘어온것같다.

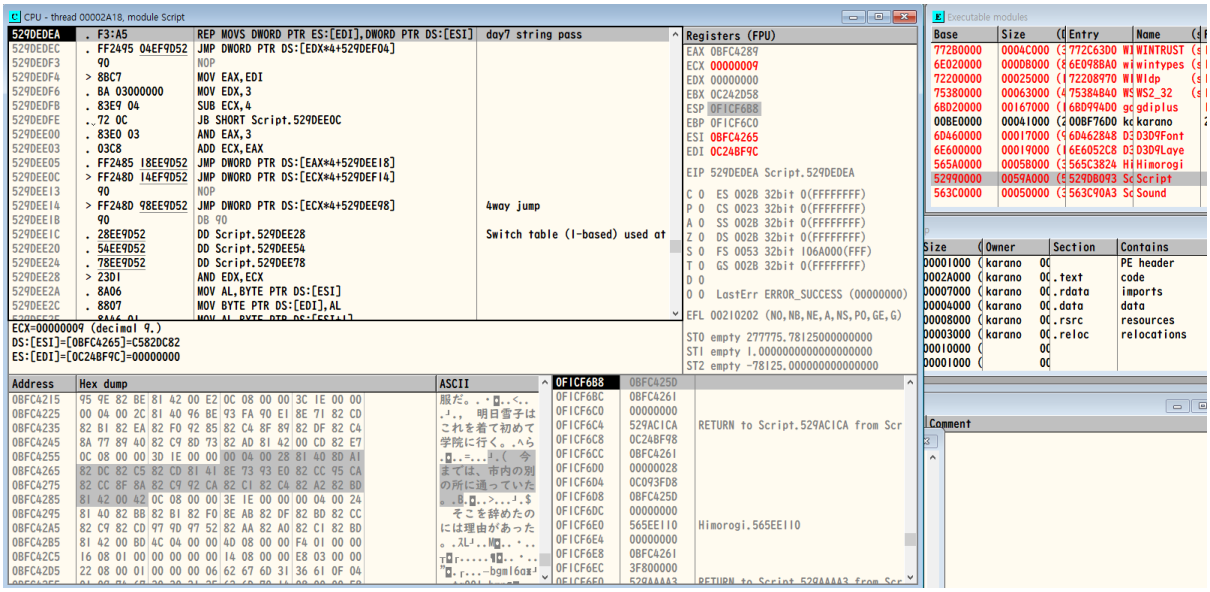
7번째 코드파인딩때 코멘트를 남겨두었던 부분이다. ESI에 있는걸 EDI가 가리키는곳에 넘기라는데..

(이 코드에만 국한된 것이 아니라 보통 ESI가 원본문자열, EDI가 복사될곳을 가리키는 식으로 많이 쓰인다고한다.)

그럼 ESI를 낚아채면 되는거 아닌가?! 라고 코드파인딩기록7에 기록되어있을듯한 기분이 든다.

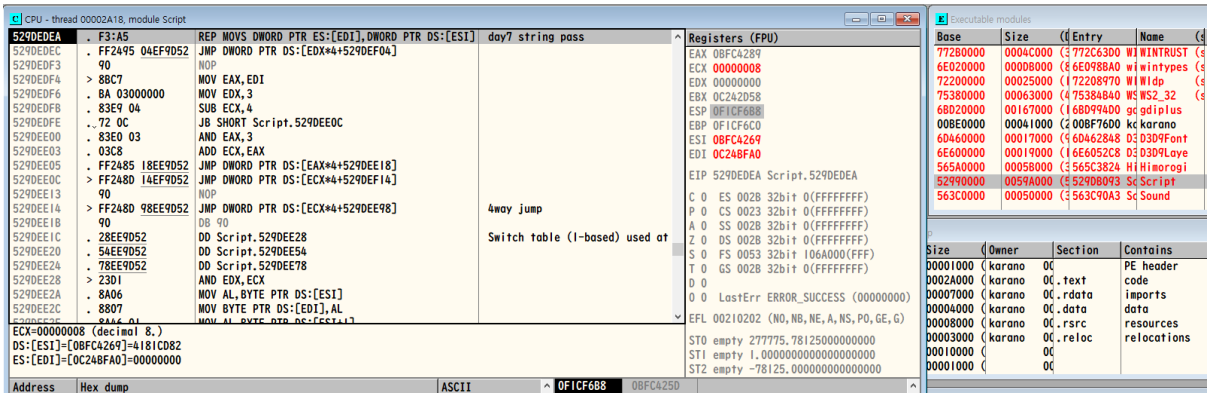
그래서 확인해봤는데 7일차에 무슨일이 있었는지 이곳에 대한 기록은 남겨져있지 않았다... 뭘까..

일단 위 추측들은 무시하고, 메모리 접근에 브포를 걸어두면 어떻게 되는지 살펴보고있었으니 재개해보도록 하자.



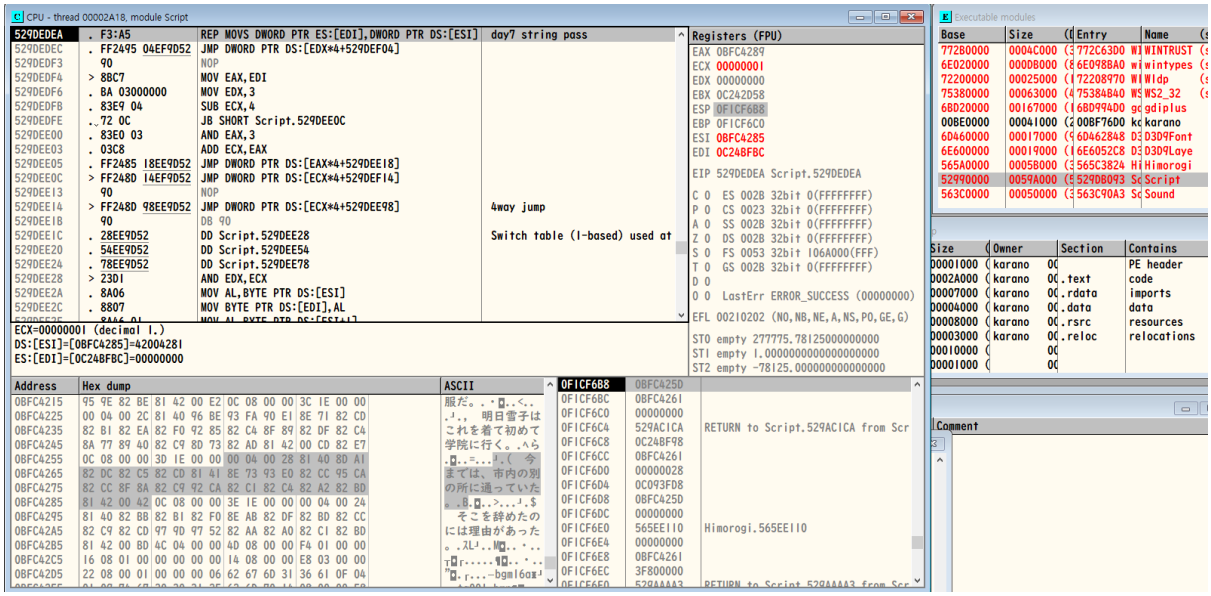
위 사진에서 F9 한번을 누르면 이렇게 된다.

ESI,EDI가 0x04증가했고, ECX가 0x01 감소했다.



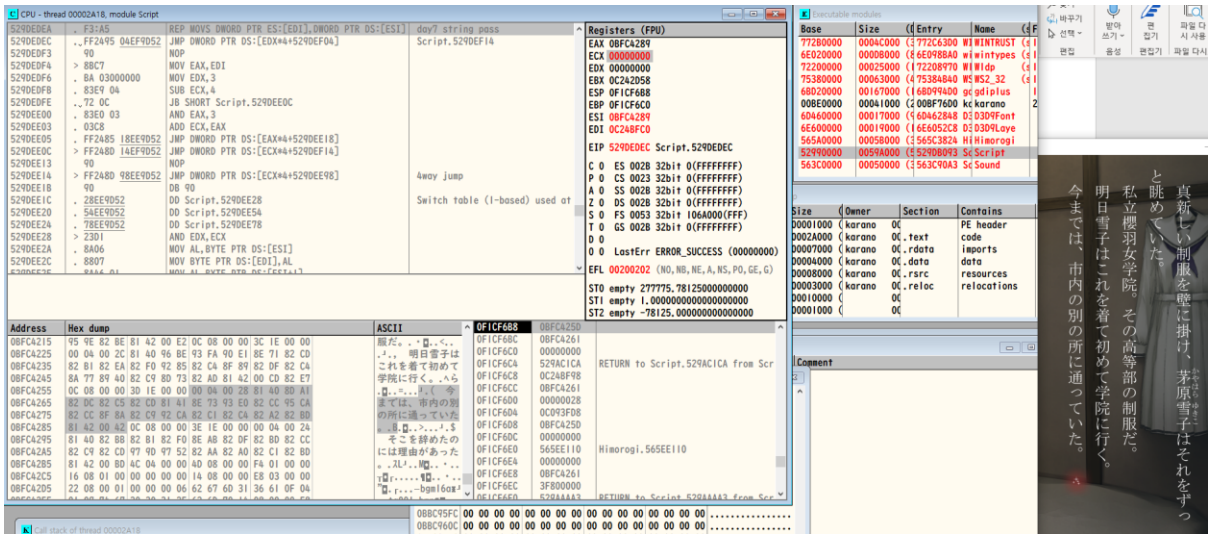
그 다음은 이렇게 또 ESI,EDI,ECX가 위와 동일하게 변화한다.

계속 진행해본다면 위 변화가 계속 반복하는 것을 확인할수 있다.



ECX가 0이 되는 부근이 중요해보이므로 0x01이 되었을 때 멈추고 캡처를 했다.

F9를 눌러 넘겨보자.



기록을 남기지 않으면 후회할뻔했다. 1에서 0으로 넘어갈때는 메모리 엑세스 없이 진행된다. 아마도 반복문 구조가 {ECX가 0이면 break; 아니면 메모리엑세스해서 잇쨔잇쨔}하는 식으로 되어있나보다. 그렇게 반복문을 나오고 나서는 브레이크 되는 지점 없이 Running이 유지되며 화면에는 방금 ESI에서 EDI로 열심히 넘겼던 대사가 출력되어있다.

이것으로 알수있는게 뭐였을까 싶긴 한데... 아무튼 그렇습니다.

오늘의 결론

529AC137 – 52990000 = 0001C137 에서 [ESP+20]를 납치하는것으로는 대사를 dumptext에 띄우는것 조차 불가능하였다. 같은 주소에서 다른 여러가지도 테스트해보긴 했으나, 뭘 추가해봤는지 기억이 잘 안나지만 모두 실패했었다..

(from 기록1~3) 0001C212에서 EDX를 납치해왔을때는 화면에서 한글출력까지 이루어졌으나, 메모리 덮어씌우기를 사용하였기때문에 버퍼크기를 넘어가는 번역문이 생긴다면 크래시가 나기 때문에 실사용이 불가능하였다.

이번에 테스트한 부분은 1C137이고 (메모리덮어씌우기 이긴 했으나) 화면출력을 성공한건 1C212이라는 점을 보았을 때.. 1C137~1C212 중간의 어떤 부분을 잘 찾아보면 되지 않을까싶다.

오늘 테스트해본지점보다는 더 깊이 들어가고, 1C212보다는 바깥에 있는 적당한 지점에서 무언가 이루어질수 있을것만 같은 기분...

추가) Frame buffer쪽은 건드리지 않는 것이 좋다는 투컨씨의 의견이 있었으니 프레임버퍼 업데이트를 하며 화면에 직접 출력하는 부분을 파보는 것은 나중에 해보겠다.