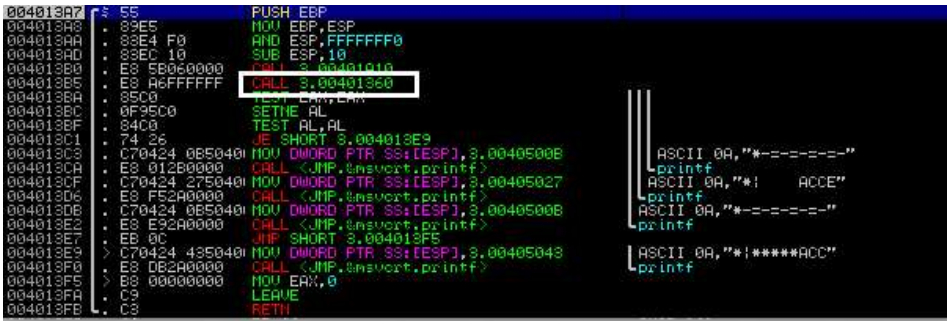
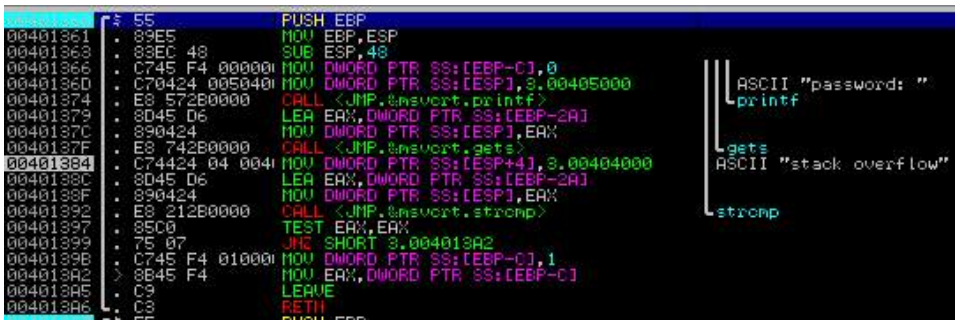
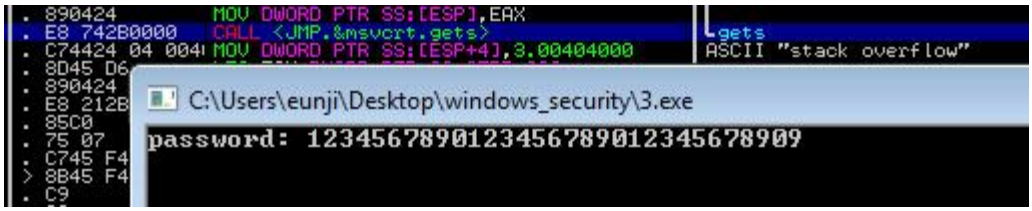
가장 먼저 디버거에서 main 함수를 찾고, main 함수에서 check_password() 호출 부분을 찾는다.



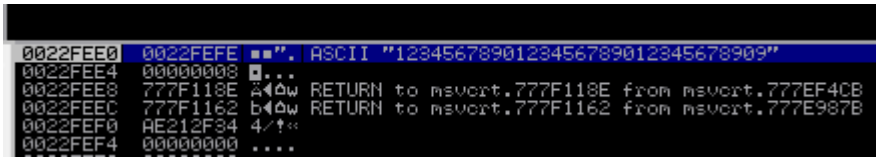
함수 프롤로그 부분부터 check_password() 함수인 것을 확인할 수 있다.

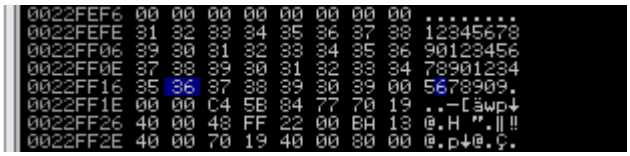


get()을 통해 사용자에게 password 입력받는 부분에 buffer[] 크기인 30보다 긴 31글자를 입력하였다.



내가 입력한 값이 저장된 곳을 스택 창에서 확인 후, dump 창에서 해당 주소로 가봤다. 가서 저장된 위치의 주소를 잘 기억해둔다. (0022FEE0 ~ 0022FF15)



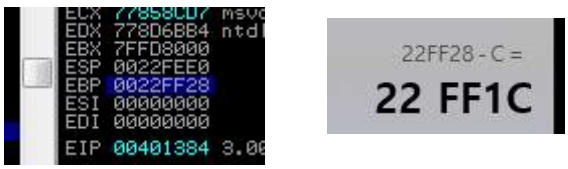


strcmp() 전 00404000에 있는 값을 [ESP+4]로 MOV 해오는 것을 알 수 있다.
이는 saved_password인 것을 알 수 있다. 주석으로 stack overflow라는 것을 알 수 있지만 덤프에서 해당 위치의 값을 확인해 보겠다.

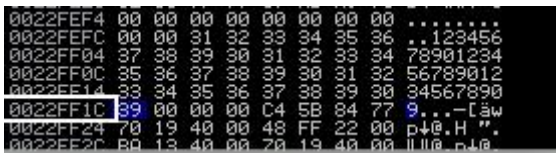
22FEE4의 위치에 saved_password 값이 저장되어 있는 것을 확인할 수 있다.

따라서 위에서 확인한 saved_password와 내가 입력한 값이 다르기 때문에
strcmp()의 결과 password가 일치하지 않아 SS[EBP-C]를 1로 mov 해주는 명령을 실행하지 않고 건너뛰게 된다.

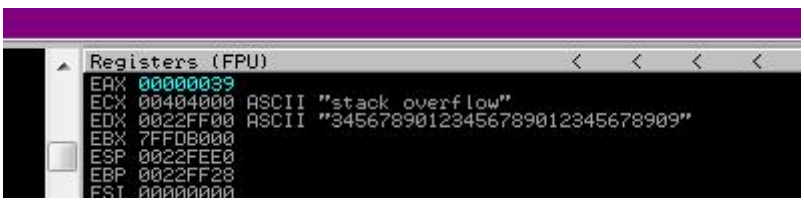
그리고 흐름상 [EBP-C]가 flag가 저장되는 위치인 것을 알 수 있다.
따라서 해당 위치를 계산해 보겠다.



EBP 값인 0022FF28에서 C를 빼면 22FF1C인 것을 알 수 있다.
덤프 창에서 22FF1C 주소로 가보겠다.

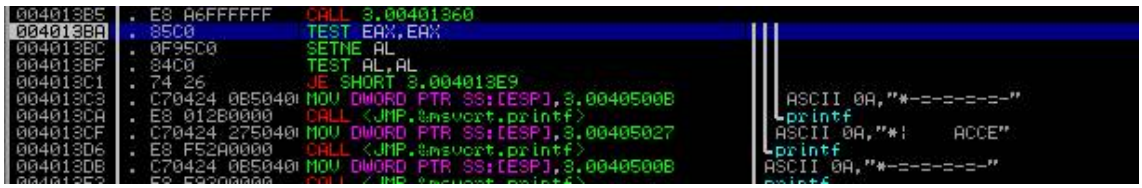


해당 위치를 확인해 보니 아까 내가 입력한 값 바로 밑에 저장된 것을 알 수 있다.
buffer의 크기가 30인데 나는 31글자를 입력해서 buffer 바로 밑인 flag가 저장되는 곳에 9가 저장된 것을 알 수 있다. 즉 flag는 9로 리턴 되게 된다.



해당 서브루틴 종료 후 다시 main 함수로 돌아오면 eax 값이 39인 상태에서

test eax, eax를 하게 된다. 그 결과 eax 값은 0이 아니라 ACCESS ALLOWED를 출력하고 끝나게 된다.



오버플로우 공격 성공 사진