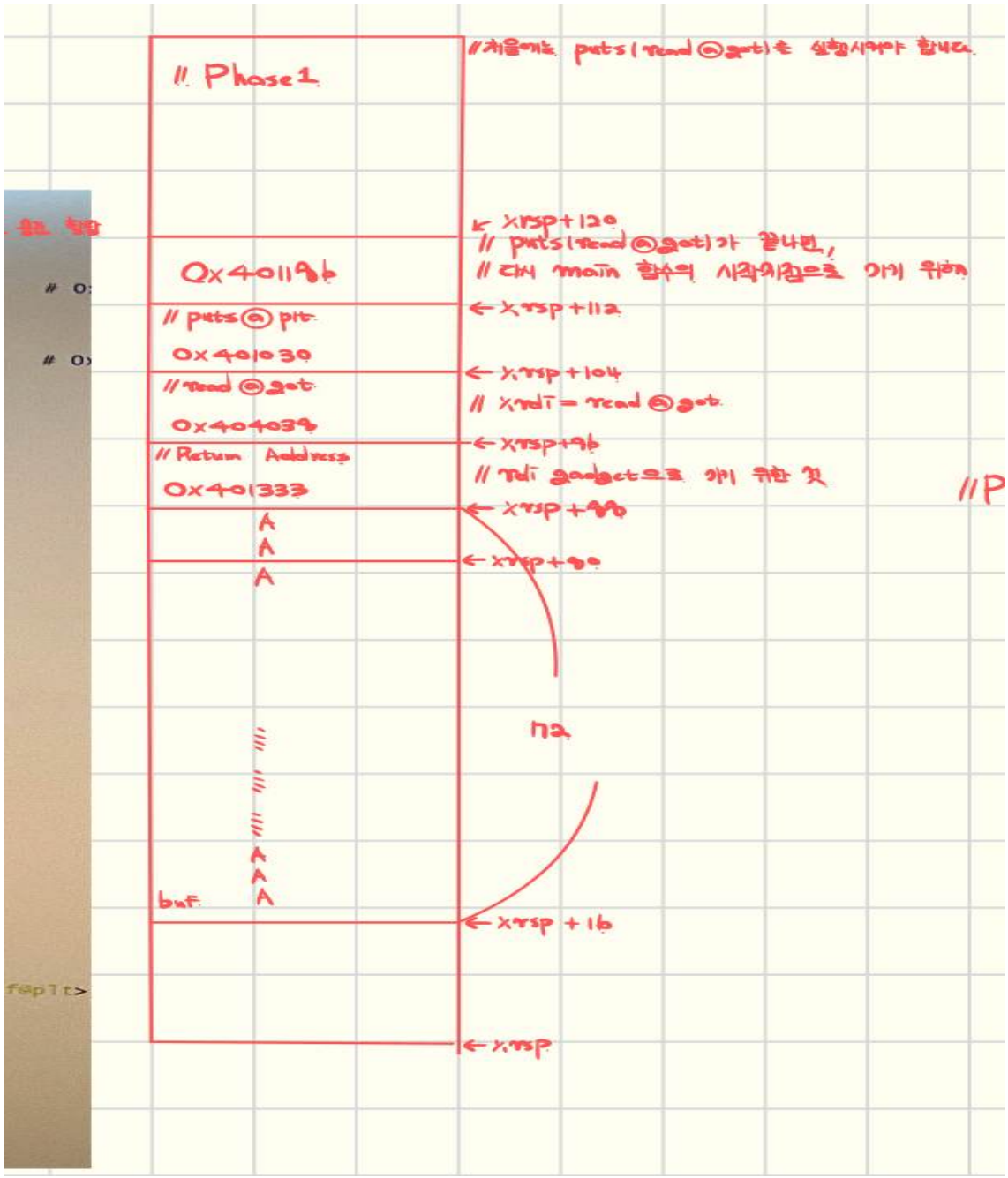
해킹및정보보안 Lab3 보고서

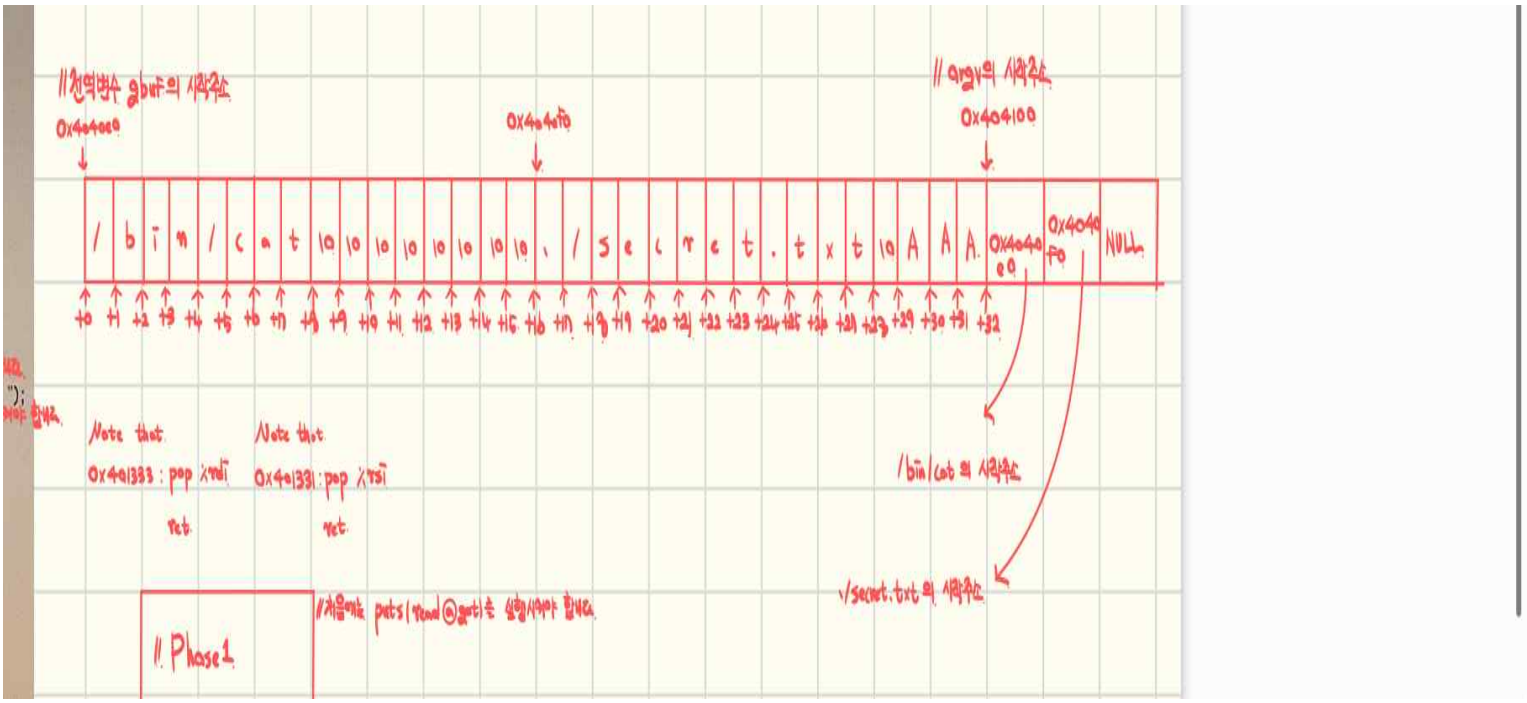
전공: 수학과 학년: 4학년 학번: 20191274 이름: 장유빈

3-2.

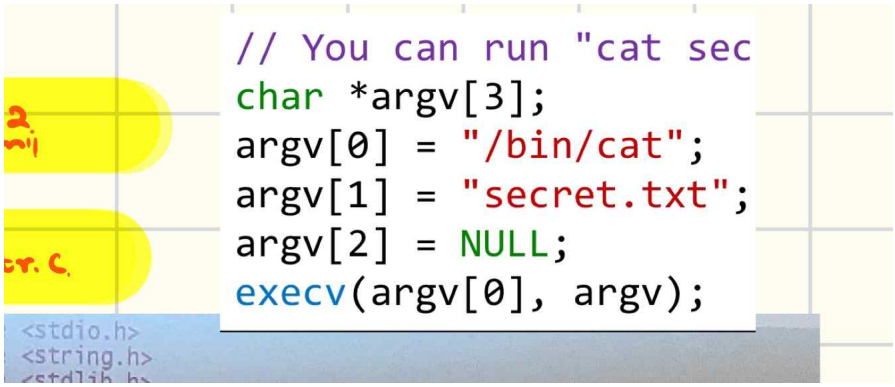
이 문제를 풀기 위해서 두 단계에 즉, 두 Phase에 걸쳐서 지역변수 buf에 BOF를 일으켜서 Control Hijack을 일으켰습니다. 먼저, 첫 단계는 “puts(read@got);”을 실행시키고 다시 main() 함수의 시작 지점으로 돌아오기 위한 BOF입니다. Stack Frame을 보면서, 자세히 설명하도록 하겠습니다.



먼저, buf의 시작과 Return Address가 저장되있는 곳까지의 거리가 10진수로 72이므로, 문자열 “A”로 채워서 원래의 Return Address까지 도달합니다. 그런다음, rdi gadget으로 가기 위해 0x401333로 채워줍니다. 다음 8바이트는 “rdi = read@got”로 설정하기 위해, 0x404038로 채워줍니다. ret 처리를 할 때, puts() 함수를 실행시키기 위해서, puts@plt의 주소인 0x401030으로 채워줍니다. 일단, 이렇게 작성한다면, “puts(read@got);”를 실행시킬 것입니다. 향후 출력된 문자열을 적절히 처리하여, base address를 구하고, 여러 offset들을 통하여, execv의 실제 주소를 구할 수 있게 됩니다. 이는 코드에 상세하게 나와있습니다. 여기서 또 중요한 점은 puts() 함수가 끝나서 ret 처리를 할 때, 우리는 아직 execv를 실행시키지 못했으므로, 그 위 8바이트 자리에 main() 함수의 시작 주소인 0x401186을 또 넣어주어야 한다는 점입니다. 일단 여기까지가 Phase 1입니다. 여기까지 정상적으로 완료했다면, 앞서 말했듯이 우리는 execv의 실제 주소를 알 수 있게 됩니다. 그리고 다시 프로그램이 끝나지 않고, main() 함수의 시작 지점으로 돌아오게 됩니다.
이제 중요한 “execv(“/bin/cat”, argv);“를 실행시키기 위한 Phase2 단계를 설명하도록 하겠습니다.
일단 스택 프레임은 뒤에서 보여드리도록 하겠습니다. main() 함수의 시작 지점으로 돌아왔기 때문에, 다시 88바이트 만큼 땡겨지게 됩니다.
우리는 전역변수 gbuf를 활용해야 합니다. 시작주소가 0x4040e0인 gbuf에 execv에 필요한 인자들을 쌓아야하기 혹은 채워야 하기 때문입니다.
일단, ”read(0, gbuf, 128):“이 실행될 때 다음과 같이 execv에 필요한 인자들을 쌓아줍니다.

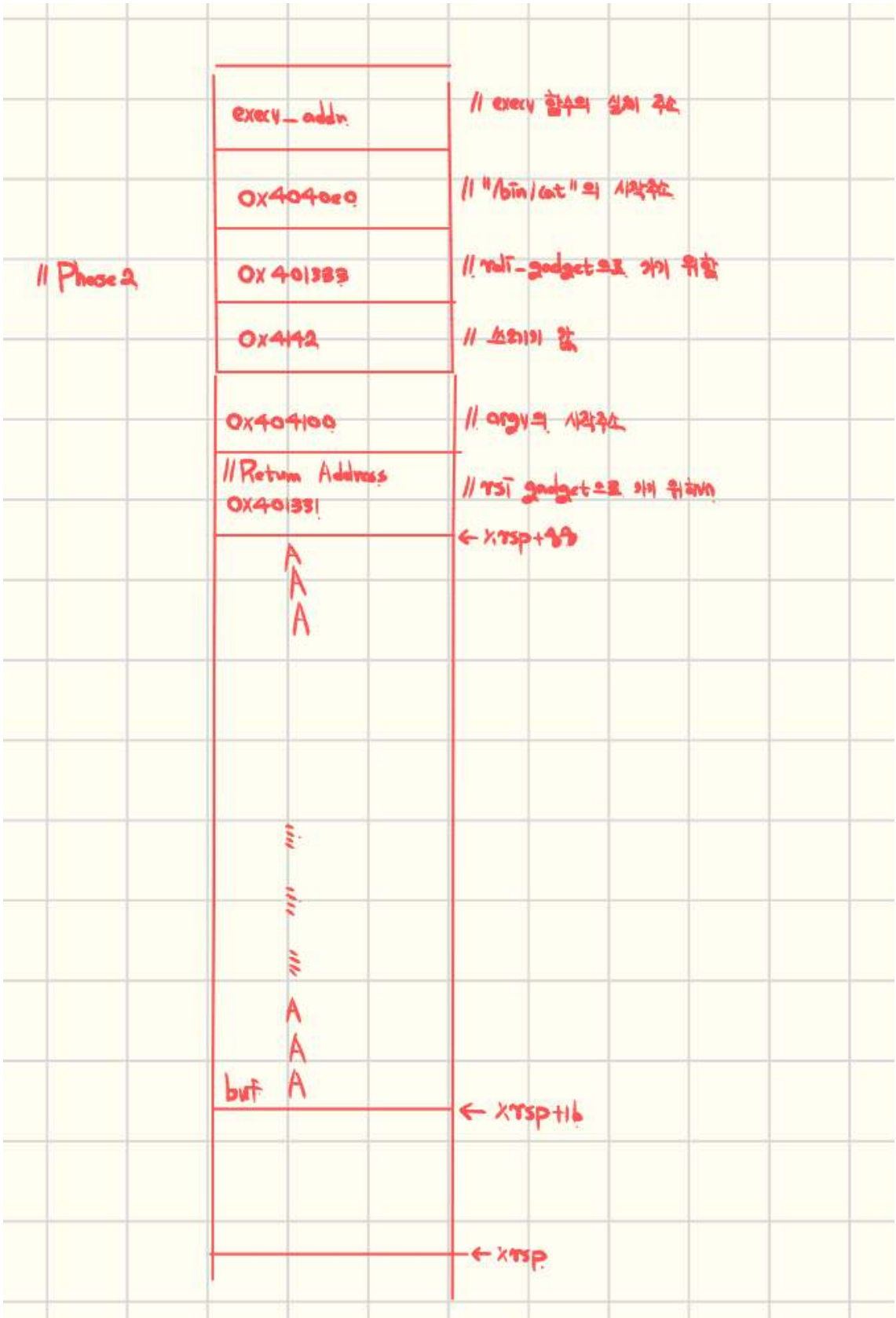


또한, char* argv[3]의 시작주소가 8의 배수로 align되게 설정하였습니다. char*의 크기는 8바이트이기 때문입니다. 이렇게 함으로써, ppt에서 설명하신 argv 배열을 정상적으로 쌓을 수 있게 됩니다. 사진은 다음과 같습니다.



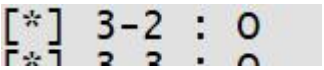
이제 "read(0, buf, 128);"에서 다시 한번 BOF를 일으켜야 합니다. 그래서 Control Hijack을 일으켜야 합니다. 먼저 Stack Frame은 다음과 같습니다.

-->> "다음 장에 계속됩니다."



똑같이 buf와 Return Address 사이를 문자열 "A"로 채워줍니다. 그리고 이번엔 rsi gadget

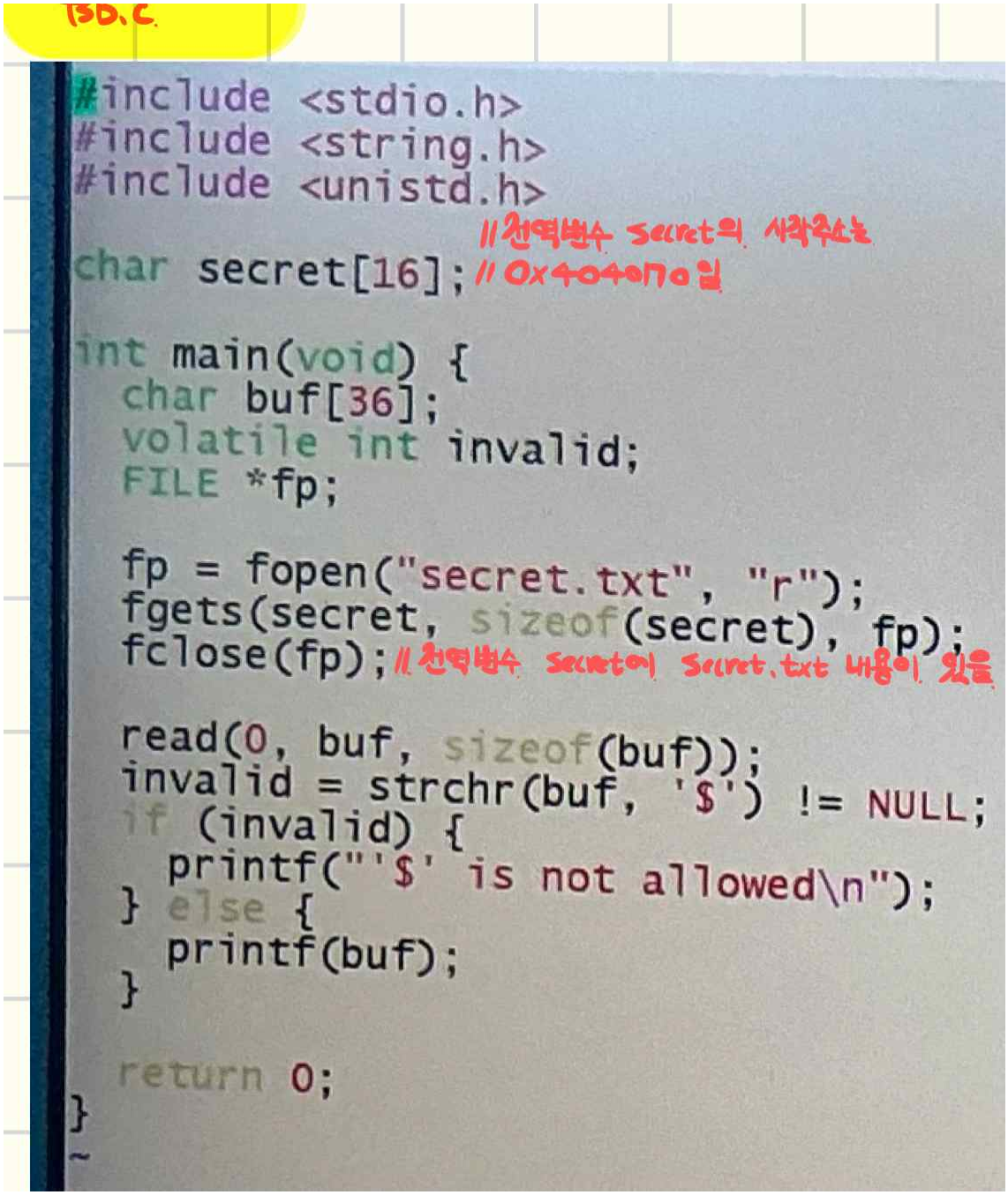
으로 가기 위해서 0x401331으로 채워줍니다. 그다음, "rsi = argv"로 설정하기 위해 argv의 시작주소인 0x404100으로 채워줍니다. r15는 아무 쓰레기값으로 채워줍니다. 그다음, ret 처리를 할 때, rdi gadget으로 가기 위해서, 0x401333으로 갑니다.
그런다음 "pop rdi"를 할 때, "rdi = "/bin/cat"로 설정하기 위해서, "/bin/cat"의 시작주소인 0x4040e0으로 채워줍니다. 이제 execv 실행에 필요한 인자인 rdi, rsi register를 다 설정 완료하였습니다. 이제 첫 번째 Phase1에서 구한 execv의 실제 주소인 execv_addr을 다음 8 바이트에 채워줍니다. ret 처리를 할 때, execv의 함수의 시작주소로 뛸 것입니다.



이렇게 하면, 3-2을 성공할 수 있습니다.

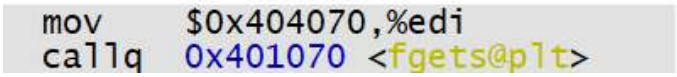
3-3.

fsb.c 소스코드는 다음과 같습니다.



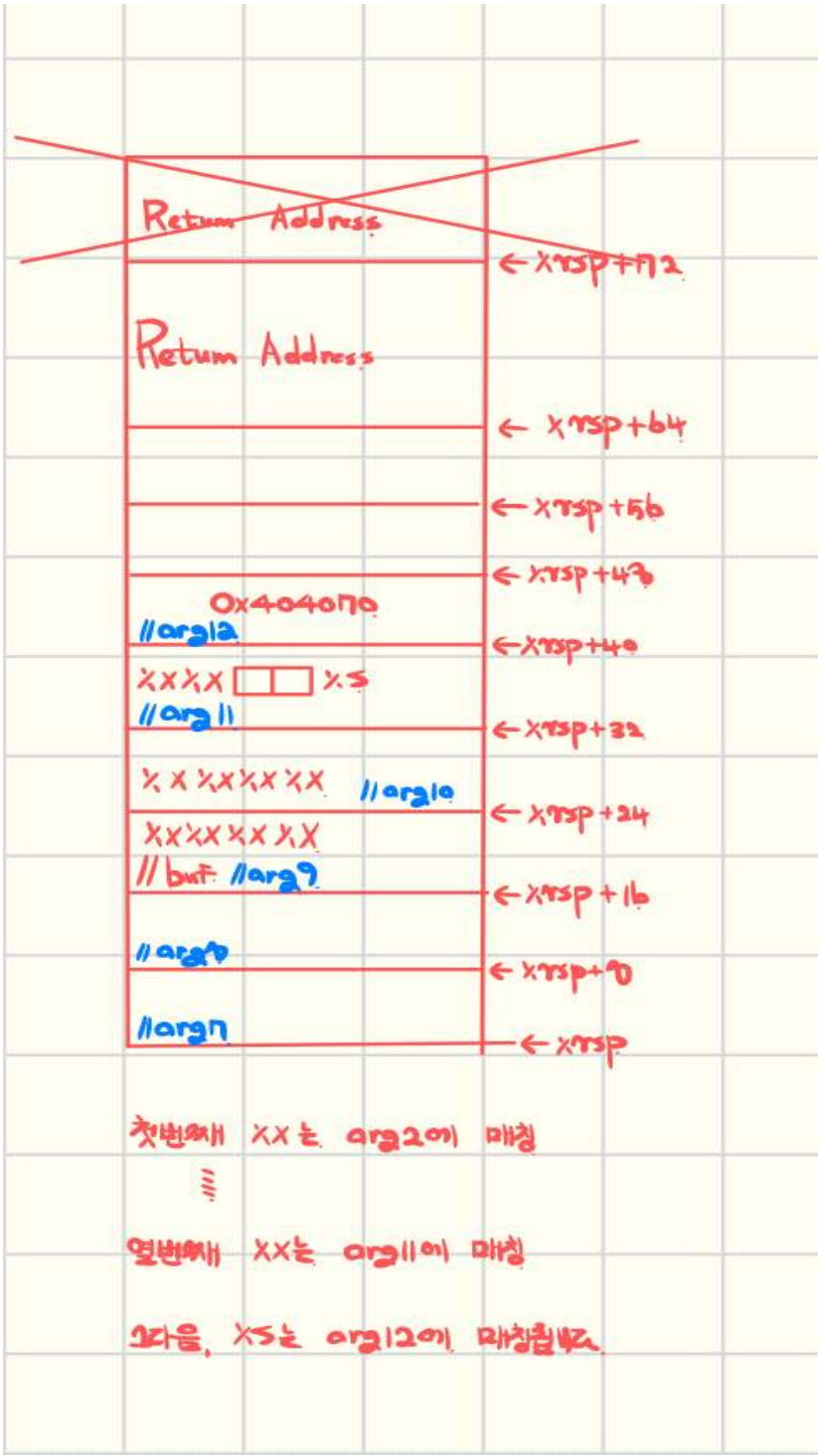
먼저, 짚고 넘어가야할 것은 fsb.c 소스파일에 보시다시피, 전역변수 secret 배열에

secret.txt의 내용이 들어간다는 것입니다. 그래서 우리는 전역변수 secret의 내용을 disclose 해야 합니다.



전역변수 secret의 시작주소는 0x404070인 것을 알 수 있습니다.
“read(0, buf, sizeof(buf));“ 코드를 통해 제가 채운 내용을 Stack Frame을 통해 설명하도록 하겠습니다.

-->> “다음 장에 계속됩니다.”



(여기 그림에서 네모칸은 띄어쓰기(공백)입니다..!!)

먼저, 우리는 수업시간 때 배웠듯이, arg7부터는 Stack에서 가져온다는 것을 배웠습니다. 그림에서 보시다시피 buf의 시작은 %rsp + 16부터입니다. 일단, %x%x%x%x(총 8바이트)를 arg9자리에 넣습니다. 그다음 또, %x%x%x%x를 arg10 자리에 집어넣습니다. 그다음 arg11 자리에 %x%x 일단 4바이트 넣고, 띄어쓰기 2개 즉, 2바이트 넣고, %s 1개인 2바이트 총 8

바이트를 집어넣습니다. 사진에 첨부되었듯이 첫 번째 %x는 arg2에 매칭됩니다. 마지막 10번째 %x는 arg11에 매칭됩니다. 그리고 다음, %s는 arg12에 매칭될 것입니다. 그렇다는 것은 %rsp + 40 ~ %rsp + 48 즉, arg12자리에 우리가 원하는 전역변수 secret의 시작주소를 p64() 함수를 이용하여, 0x404070을 집어넣습니다. 이렇게 되면, %s 포맷 지정자를 통해 전역변수 secret의 내용을 출력할 수 있게 됩니다.

```
cse20191274@cspro3:~/hacking/Lab3/3-3$ ./exploit-fsb.py
[+] Starting local process './fsb.bin': pid 994431
b'2402007c9471a97640125d78257825782578257825782578257825  1b19361b\n'
[*] Process './fsb.bin' stopped with exit code 0 (pid 994431)
```

결과물을 보시다시피, 제가 삽입한 띄어쓰기 2개를 기준으로, 앞에 것들은 %x에 해당하는 것들이고, 띄어쓰기 뒤에 내용이 secret.txt의 내용입니다.

항상 좋은 문제 감사드립니다.
이상입니다.
-20191274 장유빈-