시스템 프로그래밍

- Bomb Lab -

2014. 10. 23.

박시형

sihyeong@cnu.ac.kr

Embedded System Lab. Computer Engineering Dept.

Chungnam National University

개요

- 실습 명
  - Bomb Lab
- 목표
  - 지금까지 배운 내용을 정리하여 Bomb을 해체
- 내용
  - objdump 사용 방법
  - gdb 사용 방법
  - Bomb Lab
    - 소개
    - 주의 사항
    - Download
    - 동작 구조
    - 진행
    - 폭탄 해체 (objdump, gdb)
    - 풀이방법
  - 점수 계산
  - 제출 방법
  - 보고서 양식

objdump 사용 방법

- objdump
  - 라이브러리, 컴파일 된 오브젝트 모듈, 공유 오브젝트 파일, 독립 실행파일 등의 바이너리 파일들의 정보를 보여주는 프로그램. ELF 파일을 어셈블리어로 보여주는 Disassembler로 사용될 수 있다.
- 따라 하기
  - /home/syspro/hello.c 를 복사 후, gcc로 컴파일 하여 hello.o 실행 파일을 생성한다.
  - objdump를 사용하여 오브젝트 파일을 Disassemble 한다.

명령어 실행

```
[c000000000@eslab week6]$ gcc -c hello.c
[c000000000@eslab week6]$ objdump -d hello.o

hello.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 e4 f0                and    $0xfffffff0,%esp
   6:   83 ec 10                sub    $0x10,%esp
   9:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  10:   e8 fc ff ff ff          call   11 <main+0x11>
  15:   b8 00 00 00 00          mov    $0x0,%eax
  1a:   c9                      leave
  1b:   c3                      ret
```

실행 결과

objdump 사용 방법

- 단순히 objdump만을 사용하는 경우 순수한 어셈블리만 나와서 다소 보기가 힘들다. 따라서 C 소스코드 형태로 출력하여 보자.
- 아래 명령어는 C 소스코드 형태로 출력하게 해주는 명령이다.
  - 해당 명령어를 사용하기 위해서는 바이너리를 컴파일 할 때, **-g** 옵션을 주어 디버깅 심벌을 삽입해야 한다.

명령어 실행

실행 결과

```
[c000000000@eslab week6]$ gcc -g -c hello.c
[c000000000@eslab week6]$ objdump -S hello.o

hello.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
#include <stdio.h>

int main()
{
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 e4 f0                and    $0xfffffff0,%esp
   6:   83 ec 10                sub    $0x10,%esp
        printf("hello World\n");
   9:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  10:   e8 fc ff ff ff          call   11 <main+0x11>
        return 0;
  15:   b8 00 00 00 00          mov    $0x0,%eax
}
  1a:   c9                      leave
  1b:   c3                      ret
```

objdump 사용 방법

❖ objdump에서 사용하는 옵션들은 다음과 같다.

옵 션	설 명
a	아카이브의 헤더 정보를 보임
x	바이너리의 모든 헤더의 정보를 보임
d	실행 가능한 코드부분을 어셈블리로 출력
D	모든 부분을 어셈블리로 출력
S	소스코드와 어셈블리를 같이 출력
t	심볼 테이블을 출력
T	동적 심볼을 출력
r	재배치 엔트리를 출력
R	동적 재배치 엔트리를 출력

gdb 사용 방법

- gdb를 이용한 레지스터 값 확인
  - gdb에서 p(print) 명령어를 이용하여 레지스터의 값을 확인 할 수 있다.
  - 사용법 : p $[레지스터 명]
- 따라 하기
  - /home/syspro/gdbTest 를 홈 디렉터리로 복사 한 후, 실행파일을 gdb로 실행시킨다.

```
(gdb) b 7
Breakpoint 1 at 0x80483d5: file gdbTest.c, line 7.
```

Breakpoint 설정

```
(gdb) r
Starting program: /home/sys03/c000000000/week6/gdbTest

Breakpoint 1, main () at gdbTest.c:7
7                for(i =0; i<3; i++){
```

프로그램 실행

```
(gdb) p $eax
$1 = 1
(gdb) p $ebx
$2 = -1208205312
(gdb) p $ecx
$3 = -1970235670
```

레지스터 값 확인

gdb 사용 방법

- 레지스터 값 전체를 한번에 확인하는 명령어는 다음과 같다.
  - 사용법 : info register
    - 간단하게 i r 이라고 입력해도 동일하게 동작한다.

```
(gdb) info register
eax            0x1      1
ecx            0x8a9096ea       -1970235670
edx            0xbffff644       -1073744316
ebx            0xb7fc4000       -1208205312
esp            0xbffff5f0       0xbffff5f0
ebp            0xbffff618       0xbffff618
esi            0x0      0
edi            0x0      0
eip            0x80483d5        0x80483d5 <main+17>
eflags         0x286    [ PF SF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
```

Bomb LAB

조교의 지시 전에 '절대' 시작하지 마세요.

Bomb Lab - 소개

- Bomb Lab은 여러 단계로 이루어진 프로그램이다.

- 각 단계마다 폭탄을 해체할 수 있는 암호를 입력해야 한다.
  - 만약 여러분이 정확한 암호를 입력한다면 해당 구문의 폭탄은 해체되며, 다음 단계로 넘어간다.
  - 반면에 입력한 암호가 틀리면, 폭탄이 터지고 화면에 "BOOM!!!"이라는 메시지가 출력되고 프로그램이 종료된다.

- 모든 단계에서 정확한 암호를 입력해야 해당 폭탄이 완벽하게 해체된다.

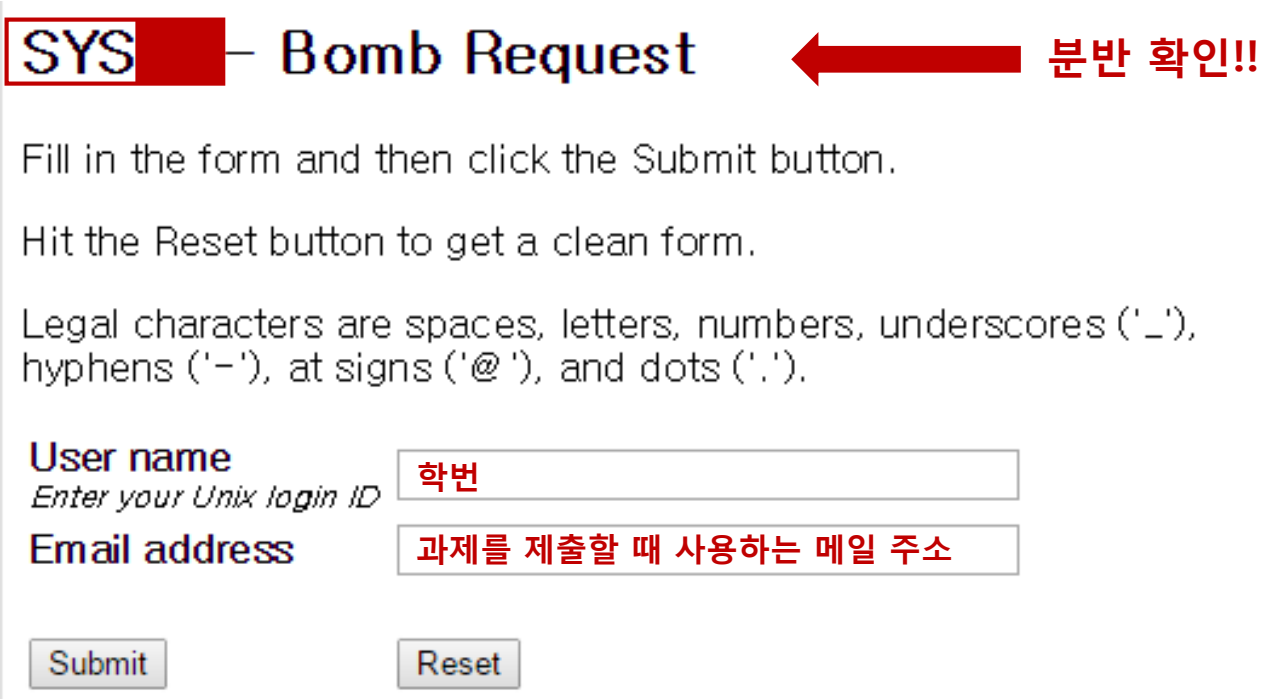
Bomb Lab - 주의 사항

- 본 프로그램은 168.188.127.145 서버에서만 동작하도록 설정되어 있습니다.
- 모든 사람의 폭탄해체 암호는 프로그램에 의해 각기 다른 방식으로 생성됩니다.
- 모든 상황은 서버를 통해 모니터링 되므로 부정행위의 소지가 있는 행동에 각별히 주의바랍니다.
  - 부정한 방법(바이너리 해킹 등)으로 해체 시도 시, 0점 처리됨과 동시에 기존 과제 모두 0점
- 다운 받은 Bomb 파일의 관리소홀로 인해 삭제될 경우, 복구가 불가하여 0점 처리될 수 있으니 주의하시길 바랍니다.
- Bomb의 해체/폭발 정보는 자동으로 서버로 전송되어 점수가 계산됩니다.
- 반드시 하나의 폭탄만을 다운 받으세요. (두 개 이상의 폭탄을 다운받은 자는 copy로 간주하고 0점 처리)
- 주의사항을 위반하여 발생한 문제에 대해서는 각자가 책임지는 것을 원칙으로 합니다.

Bomb Lab - Download

❖ Bomb 받는 방법

- http://168.188.127.145:{포트번호} 에 접속
- 해당 주소가 자신의 분반과 맞는 주소인지 확인
- 자신의 학번과 이메일 주소를 정확하게 입력한 후 Submit을 클릭
  - 이때, 여러 사람이 동시에 접속하기 때문에 반응이 느릴 수 있음.
  - Submit 버튼은 한번만 누르기! (여러 번 누르면, 폭탄이 여러 개 받아짐)

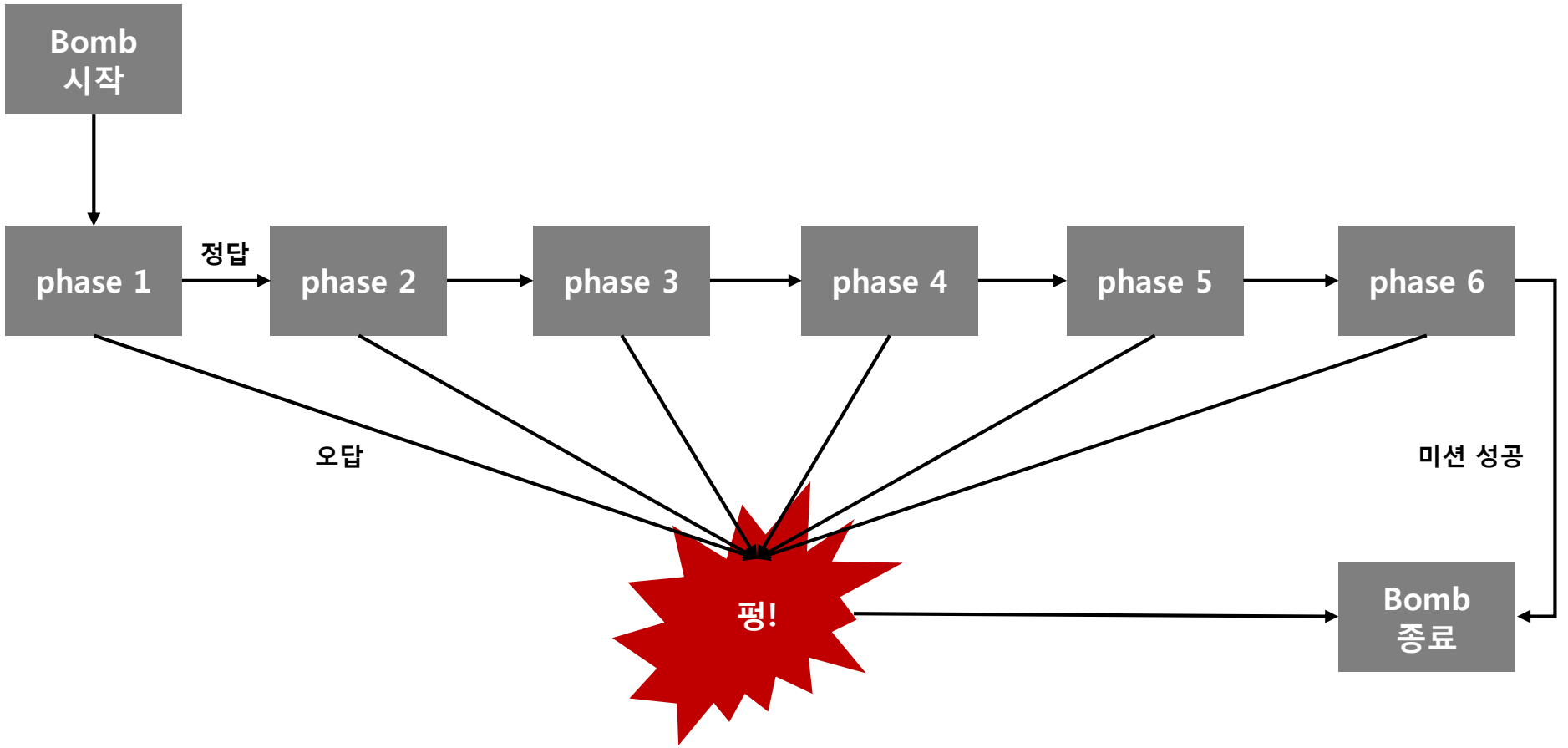


Bomb Lab - Download

- Bomb을 계정에서 실행
  - 다운받은 bombX.tar 파일을 WinSCP를 이용하여 자신의 계정으로 옮긴다.
  - bombX.tar 파일을 압축 해제한다.
    - tar xf bombX.tar

- 수업시간에 Bomb을 다운받지 못한 학생은 공대5호관 533호(임베디드 시스템 연구실)로 방문 하세요.

Bomb Lab - 동작 구조



Bomb Lab – 진행

- bomb.c 파일을 확인하여 폭탄의 내용 구성을 확인한 후, 도구들을 이용하여 bomb 파일을 분석하고 암호를 알아내야 한다.
- ./bomb을 수행시키면 각 단계별로 암호를 입력 하여 다음 단계로 갈 수 있으며, 다음과 같이 한꺼번에 입력할 수도 있다.
  - shell> **./bomb solution.txt**
    - solution.txt는 각 단계별 정답을 enter로 구분하여 입력한 파일
    - gdb 에서도 사용 가능하다.
      - gdb bomb solution.txt
- 실행 후 ctrl+c 명령을 이용해서 취소가 가능하므로 실수로 폭탄이 터지지 않도록 주의 요망!

Bomb Lab - 폭탄 해체

- 폭탄을 해체 하는 방법은 2가지가 있다.
  - objdump
    - 소스코드의 구조를 이해하는데 도움을 줄 수 있다.
  - gdb
    - 프로그램이 실행되면서 변화하는 값과 프로그램의 동작과정을 확인 할 수 있다.
  - 가장 좋은 방법은 상황에 맞추어 두 가지 방법을 모두 사용하는 것이 좋다.
  - 폭탄을 어떻게 해체하는지에 대한 방법을 이해시키기 위해 다음의 1단계의 폭탄 암호를 해체하는 과정을 통해 익히도록 한다.

Bomb Lab – 폭탄 해체 (objdump)

- objdump 를 사용해서 코드를 어셈블리어로 바꾼다.

```
[bOOOOOOOOO@eslab bomb2]$ objdump -S bomb > dump.txt
[bOOOOOOOOO@eslab bomb2]$ ls
README  bomb  bomb.c  dump.txt

08048a00 <main>:
 8048a00:       55                      push   %ebp
 a048a01:       89 e5                   mov    %esp,%ebp
 8048a03:       53                      push   %ebx
 8048a04:       83 e4 f0                and    $0xfffffff0,%esp
 8048a07:       83 ec 10                sub    $0x10,%esp
 8048a0a:       8b 45 08                mov    0x8(%ebp),%eax
 8048a0d:       8b 5d 0c                mov    0xc(%ebp),%ebx
 8048a10:       83 f8 01                cmp    $0x1,%eax
 8048a13:       75 0c                   jne    8048a21 <main+0x21>
 8048a15:       a1 c4 c7 04 08          mov    0x804c7c4,%eax
 8048a1a:       a3 ec c7 04 08          mov    %eax,0x804c7ec
 8048a1f:       eb 64                   jmp    8048a85 <main+0x85>
 8048a21:       83 f8 02                cmp    $0x2,%eax
 8048a24:       75 41                   jne    8048a67 <main+0x67>
 8048a26:       c7 44 24 04 6c a1 04    movl   $0x804a16c,0x4(%esp)
```

- 생성된 dump.txt파일을 vi 에디터를 통해서 열어보면 bomb 파일이 어셈블리어로 변환된 것을 볼 수 있다.
- 이를 통해서 전체 프로그램의 구조를 분석해 나갈 수 있다.

Bomb Lab – 폭탄 해체 (objdump)

- 가장 먼저 어셈블리어 코드를 분석해서 오답이 입력될 경우 폭탄을 터트리는 부분을 찾아야 한다.

```
 8048b77:       85 c0                   test   %eax,%eax
 8048b79:       74 05                   je     8048b80 <phase_1+0x20>
 8048b7b:       e8 15 07 00 00          call   8049295 <explode_bomb>

08049295 <explode_bomb>:
 8049295:       83 ec 1c                sub    $0x1c,%esp
 8049298:       c7 04 24 7d a5 04 08    movl   $0x804a57d,(%esp)
 804929f:       e8 3c f5 ff ff          call   80487e0 <puts@plt>
 80492a4:       c7 04 24 86 a5 04 08    movl   $0x804a586,(%esp)
 80492ab:       e8 30 f5 ff ff          call   80487e0 <puts@plt>
 80492b0:       c7 04 24 00 00 00 00    movl   $0x0,(%esp)
 80492b7:       e8 05 ff ff ff          call   80491c1 <send_msg>
 80492bc:       c7 04 24 34 a4 04 08    movl   $0x804a434,(%esp)
 80492c3:       e8 18 f5 ff ff          call   80487e0 <puts@plt>
 80492c8:       c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 80492cf:       e8 2c f5 ff ff          call   8048800 <exit@plt>
```

- 각 단계마다 어떤 값을 비교하는 부분과 explode_bomb 으로 점프하는 모습을 볼 수 있다. 이를 통해서 오답인 경우 explode_bomb로 이동하는 것이라 예측 할 수 있다.

Bomb Lab – 폭탄 해체 (objdump)

❖ phase_1을 보면 입력한 문자열과 다른 경우 explode_bomb를 호출하는 것을 볼 수 있다.

```
08048b60 <phase_1>:
 8048b60:       83 ec 1c                sub    $0x1c,%esp
 8048b63:       c7 44 24 04 a8 a2 04    movl   $0x804a2a8,0x4(%esp)
 8048b6a:       08
 8048b6b:       8b 44 24 20             mov    0x20(%esp),%eax
 8048b6f:       89 04 24                mov    %eax,(%esp)
 8048b72:       e8 83 04 00 00          call   8048ffa <strings_not_equal>
 8048b77:       85 c0                   test   %eax,%eax
 8048b79:       74 05                   je     8048b80 <phase_1+0x20>
 8048b7b:       e8 15 07 00 00          call   8049295 <explode_bomb>
 8048b80:       83 c4 1c                add    $0x1c,%esp
 8048b83:       c3                      ret
```

Bomb Lab – 폭탄 해체 (gdb)

- 앞에서 보였던 objdump를 이용하는 것과 마찬가지로 gdb를 통해서도 어셈블리어 코드를 확인 할 수 있다.
  - **disassemble** 명령

```
(gdb) disassemble phase_1
Dump of assembler code for function phase_1:
   0x08048b60 <+0>:     sub    $0x1c,%esp
   0x08048b63 <+3>:     movl   $0x804a2a8,0x4(%esp)
   0x08048b6b <+11>:    mov    0x20(%esp),%eax
   0x08048b6f <+15>:    mov    %eax,(%esp)
   0x08048b72 <+18>:    call   0x8048ffa <strings_not_equal>
   0x08048b77 <+23>:    test   %eax,%eax
   0x08048b79 <+25>:    je     0x8048b80 <phase_1+32>
   0x08048b7b <+27>:    call   0x8049295 <explode_bomb>
   0x08048b80 <+32>:    add    $0x1c,%esp
   0x08048b83 <+35>:    ret
End of assembler dump.
```

- 해당 명령어는 gdb에서 objdump와 같은 기능을 하는 명령어이다.
  - disassemble [함수 명] : 함수의 어셈블리 코드를 출력한다.
  - disassemble [주소 1] [주소 2] : 주소 1 ~ 주소 2 범위 사이의 어셈블리 코드를 출력한다.

Bomb Lab – Phase 1

- phase 1의 구조를 보면 strings_not_equal 함수를 호출하는 것을 볼 수 있다. 이를 통해 어떠한 문자열을 입력 받아서 비교를 한 뒤, 오답일 경우 explode_bomb을 호출한다는 것을 추측할 수 있다.

```
(gdb) disassemble phase_1
Dump of assembler code for function phase_1:
   0x08048b60 <+0>:     sub    $0x1c,%esp
   0x08048b63 <+3>:     movl   $0x804a340,0x4(%esp)
   0x08048b6b <+11>:    mov    0x20(%esp),%eax
   0x08048b6f <+15>:    mov    %eax,(%esp)
   0x08048b72 <+18>:    call   0x804909a <strings_not_equal>
   0x08048b77 <+23>:    test   %eax,%eax
   0x08048b79 <+25>:    je     0x8048b80 <phase_1+32>
   0x08048b7b <+27>:    call   0x8049335 <explode_bomb>
   0x08048b80 <+32>:    add    $0x1c,%esp
   0x08048b83 <+35>:    ret
End of assembler dump.
```

- 비교하는 값을 찾아야 하는데, 함수의 앞 부분에서 0x4(%esp)의 값에 $0x804a340이라는 값을 옮기는 것을 볼 수 있다. 이 부분의 값을 보면 아래와 같다.

```
(gdb) x/s 0x804a340
0x804a340:      "I am a boy, you are a girl"
```

Bomb Lab – Phase 1

- 앞에서 찾아낸 문자열을 입력하면 아래와 같이 다음 단계로 넘어간다.

```
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```

정답 입력

```
Phase 1 defused. How about the next one?
```

- 오답일 경우 아래와 같이 폭탄이 터지게 된다.

```
which to blow yourself up. Have a nice day!
```

오답 입력

```
BOOM!!!
The bomb has blown up.
Your instructor has been notified.
```

Bomb Lab – 풀이 방법

- 결국 각 단계의 폭탄을 해체하기 위한 암호를 찾기 위해서는 어셈블리어로 변환된 **코드의 구조를 이해**해야 한다.

- 그리고 나서 의심 가는 부분을 gdb를 통해 값을 가져와 확인해 보아야 한다.

- 이를 위하여 **objdump**로 **전체코드를 분석**한 다음 **gdb**를 이용해서 **하나 하나 추적**해 나가는 과정이 필요하다.

Bomb Lab - 점수 계산

- 아래의 웹 페이지를 통해 실시간으로 각자의 진행상황(해체/폭발)을 확인 할 수 있습니다.
  - http://168.188.127.145:{포트번호}/scoreboard.htm
  - 폭탄을 모두 해체할 경우 60점
  - 폭탄이 터진 경우
    - 2회 : -1점 / 20회 : -10점

- Time Attack 방식으로 채점, 모든 폭탄(6 단계)을 제거했을 경우 완료 메일을 조교에게 보낸다.
  - sihyeong@cnu.ac.kr
  - 메일제목 : [sys02]폭탄완료_학번_이름

제출 사항

- **Bomb Lab 기간**
  - **10월 23일 ~ 11월 5일 (2주)**
  - **11월 5일 23시 59분 59초에 bomb lab 서버 종료**

- **Bomb Lab 보고서**
  - 결과화면을 붙임(폭탄해체 화면과 웹 페이지에서의 본인 결과)
  - 결과화면에 학번이 보이도록 캡쳐
  - 풀이과정에 대한 자세한 설명 (각 단계별 설명 + a)
  - Bomb Lab을 통해 느낀 점 및 에피소드, 하고 싶은 말
  - 뒷장의 보고서 양식 참조

보고서 양식

- 1. 표지
- 2. 목차
- 3. 개요
- 4. 각 단계에 대한 해결 방법 (캡쳐 포함)
  - 4.1 해결 방법
    - 단계별로 어떤 방법으로 접근하여 문제를 풀었는지 최대한 단계별로 자세히 설명
  - 4.2 Flow Chart(순서도)
    - 어셈블리 소스를 토대로 순서도를 작성
    - (순서도 없을 경우 감점)
  - 4.3 정답
- 5. 고찰 및 느낀 점