시스템 프로그래밍

Bomblab

2016.11.01

황슬아

seula.hwang@cnu.ac.kr

개요

1. 실습명
   - ✓ Bomb Lab
2. 목표
   - ✓ 지금까지 배운 내용을 정리하여 Bomb을 해체
3. 내용
   - ✓ objdump 사용 방법
   - ✓ gdb 사용 방법
   - ✓ Bomb Lab
     - 소개
     - 주의 사항
     - Download
     - 동장 구조
     - 진행
     - 폭탄 해체
     - objdump, gdb
     - 풀이 방법
   - ✓ 점수계산
   - ✓ 제출 방법
   - ✓ 보고서양식

objdump 사용 방법

1. objdump
   - ✓ 라이브러리, 컴파일 된 오브젝트 모듈, 공유 오브젝트 파일, 독립 실행파일 등의 바이너리 파일들의 정보를 보여주는 프로그램. ELF 파일을 어셈블리어로 보여주는 Disassembler 로 사용될 수 있다.

2. 아래 명령어는 C 소스코드 형태로 출력하게 해주는 명령이다.
   - ✓ 해당 명령어를 사용하기 위해서는 바이너리를 컴파일 할 때, -g 옵션을 주어 디버깅 심벌을 삽입해야 한다.

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

hello.o:     file format elf32-i386


Disassembly of section .text:

OOOOOOOO <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

3. 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

1. gdb를 이용한 레지스터 값 확인
   1) gdb에서 p(print) 명령어를 이용하여 레지스터의 값을 확인 할 수 있다.
   2) 사용법 : p $[레지스터 명]

```
(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

2. 레지스터 값 전체를 한번에 확인하는 명령어는 다음과 같다.

1) 사용법 : info register

- 간단하게 ir 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

1. Bomb Lab은 여러 단계로 이루어진 프로그램이다.

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

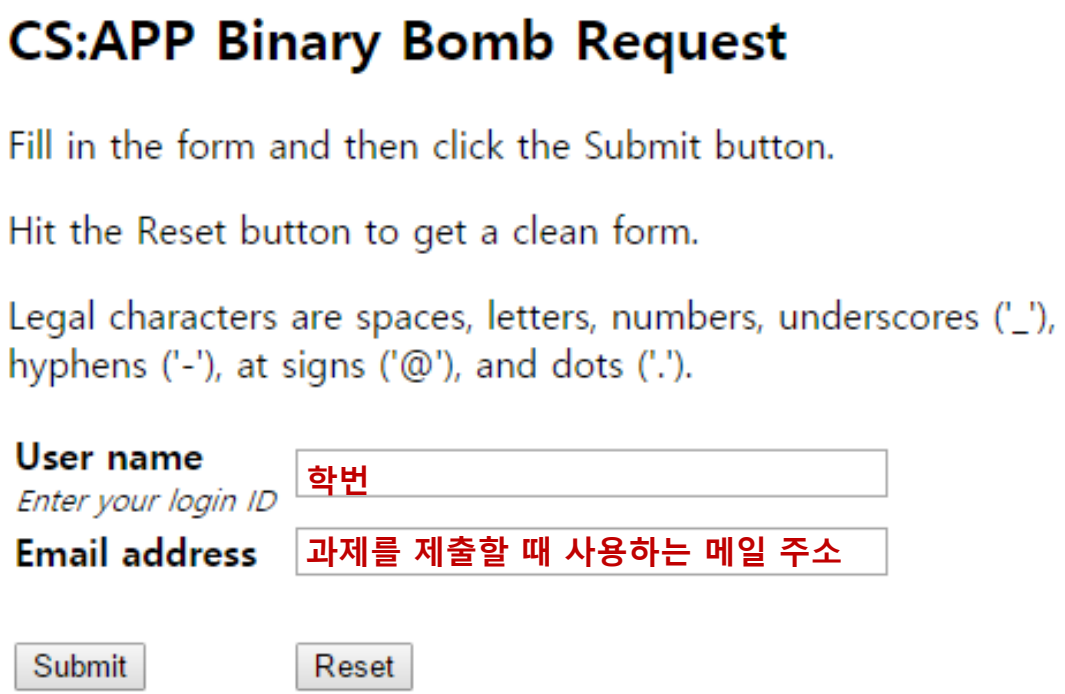
3. 모든 단계에서 정확한 암호를 입력해야 해당 폭탄이 완벽하게 해체된다.

Bomb Lab - 주의 사항

1. 본 프로그램은 133.186.135.128 서버에서만 동작하도록 설정되어 있습니다.
2. 모든 사람의 폭탄 해체 암호는 프로그램에 의해 각기 다른 방식으로 생성됩니다.
3. 모든 상황은 서버를 통해 모니터링 되므로 부정행위의 소지가 있는 행동에 각별히 주의바랍니다.
    1) 부정한 방법(바이너리 해킹 등)으로 해체 시도 시, 0점 처리됨과 동시에 기존 과제 모두 0점
4. 다운 받은 Bomb 파일의 관리소홀로 인해 삭제될 경우, 복구가 불가하여 0점 처리될 수 있으니 주의하시길 바랍니다.
5. Bomb의 해체/폭발 정보는 자동으로 서버로 전송되어 점수가 계산됩니다.
6. 반드시 하나의 폭탄만을 다운 받으세요. (두 개 이상의 폭탄을 다운받은 자는 copy로 간주하고 0점 처리)
7. 주의사항을 위반하여 발생한 문제에 대해서는 각자가 책임지는 것을 원칙으로 합니다.

Bomb Lab - Download

1. Bomb 받는 방법
   1) http://133.186.135.128:1500 에 접속
   2) 해당 주소가 자신의 분반과 맞는 주소인지 확인
   3) 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 -> **COPY로 간주**)

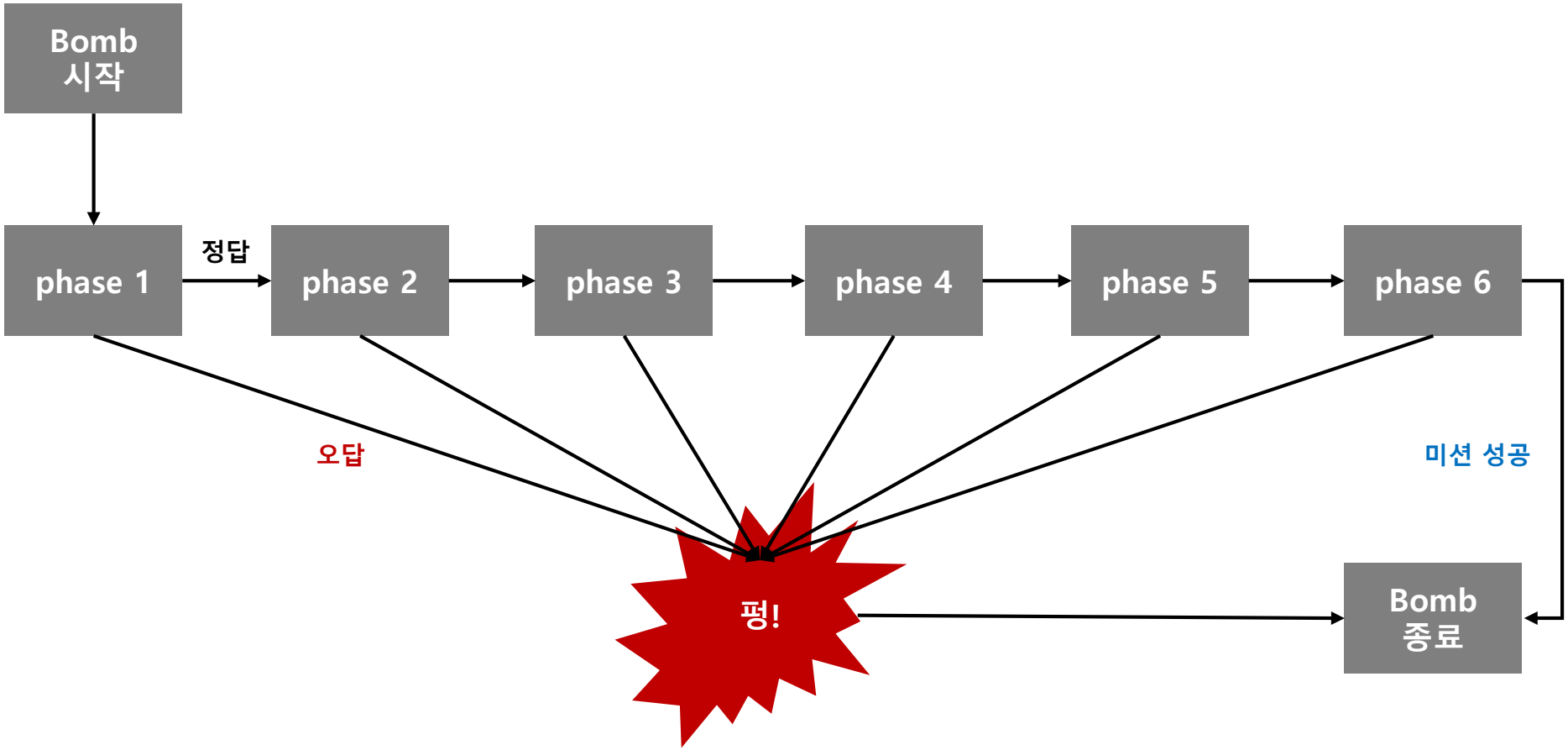


Bomb Lab - Download

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

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

Bomb Lab – 동작 구조



Bomb Lab - 진행

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

Bomb Lab – 폭탄 해체

1. 폭탄을 해체 하는 방법은 2가지가 있다.

   1) objdump
      - 소스코드의 구조를 이해하는데 도움을 줄 수 있다.

   2) gdb
      - 프로그램이 실행되면서 변화하는 값과 프로그램의 동작 과정을 확인 할 수 있다.

   3) 가장 좋은 방법은 상황에 맞추어 두 가지 방법을 모두 사용하는 것이다.

   4) 폭탄을 어떻게 해체하는지에 대한 방법을 이해시키기 위해 다음의 1단계의 폭탄 암호를 해체하는 과정을 통해 익히도록 한다.

Bomb Lab – 폭탄 해체 (objdump)

1. objdump 를 사용해서 코드를 어셈블리어로 바꾼다.

```
sysOO@localhost:~/bomblab/bomb1$ objdump -S bomb > dump.txt
sysOO@localhost:~/bomblab/bomb1$ ls
bomb  bomb.c  dump.txt  README
```

```
000000000400e4d <main>:
 */

FILE *infile;

int main(int argc, char *argv[])
{
  400e4d:       53                      push   %rbx
    /* Note to self: remember to port this bomb to Windows and put a
     * fantastic GUI on it. */

    /* When run with no arguments, the bomb reads its input lines
     * from standard input. */
    if (argc == 1) {
  400e4e:       83 ff 01                cmp    $0x1,%edi
  400e51:       75 10                   jne    400e63 <main+0x16>
        infile = stdin;
  400e53:       48 8b 05 4e 39 20 00    mov    0x20394e(%rip),%rax        # 6047
a8 <stdin@@GLIBC_2.2.5>
  400e5a:       48 89 05 5f 39 20 00    mov    %rax,0x20395f(%rip)        # 6047
c0 <infile>
  400e61:       eb 63                   jmp    400ec6 <main+0x79>
  400e63:       48 89 f3                mov    %rsi,%rbx

    /* When run with one argument <file>, the bomb reads from <file>
     * until EOF, and then switches to standard input. Thus, as you
     * defuse each phase, you can add its defusing string to <file> and
     * avoid having to retype it. */
    else if (argc == 2) {
  400e66:       83 ff 02                cmp    $0x2,%edi
  400e69:       75 3a                   jne    400ea5 <main+0x58>
        if (!(infile = fopen(argv[1], "r"))) {
```

2. 생성된 dump.txt파일을 vi 에디터를 통해서 열어보면 bomb 파일이 어셈블리어로 변환된 것을 볼 수 있다.

3. 이를 통해서 전체 프로그램의 구조를 분석해 나갈 수 있다.

Bomb Lab – 폭탄 해체 (objdump)

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

```
400fa0:       74 05                   je     400fa7 <phase_1+0x17>
400fa2:       e8 ca 06 00 00          callq  401671 <explode_bomb>
```

```
0000000000401671 <explode_bomb>:
401675:       bf 81 29 40 00          mov    $0x402981,%edi
40167a:       e8 41 f5 ff ff          callq  400bc0 <puts@plt>
40167f:       bf 8a 29 40 00          mov    $0x40298a,%edi
401684:       e8 37 f5 ff ff          callq  400bc0 <puts@plt>
401689:       bf 00 00 00 00          mov    $0x0,%edi
40168e:       e8 d2 fe ff ff          callq  401565 <send_msg>
401693:       bf 30 28 40 00          mov    $0x402830,%edi
401698:       e8 23 f5 ff ff          callq  400bc0 <puts@plt>
40169d:       bf 08 00 00 00          mov    $0x8,%edi
4016a2:       e8 49 f6 ff ff          callq  400cf0 <exit@plt>
```

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

Bomb Lab – 폭탄 해체 (objdump)

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

```
0000000000400f90 <phase_1>:
  400f90:       48 83 ec 08             sub    $0x8,%rsp
  400f94:       be 70 26 40 00          mov    $0x402670,%esi
  400f99:       e8 fa 03 00 00          callq  401398 <strings_not_equal>
  400f9e:       85 c0                   test   %eax,%eax
  400fa0:       74 05                   je     400fa7 <phase_1+0x17>
  400fa2:       e8 ca 06 00 00          callq  401671 <explode_bomb>
  400fa7:       48 83 c4 08             add    $0x8,%rsp
  400fab:       c3                      retq
```

Bomb Lab – 폭탄 해체 (gdb)

1. 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:
   0x0000000000400f90 <+0>:     sub    $0x8,%rsp
   0x0000000000400f94 <+4>:     mov    $0x402670,%esi
   0x0000000000400f99 <+9>:     callq  0x401398 <strings_not_equal>
   0x0000000000400f9e <+14>:    test   %eax,%eax
   0x0000000000400fa0 <+16>:    je     0x400fa7 <phase_1+23>
   0x0000000000400fa2 <+18>:    callq  0x401671 <explode_bomb>
   0x0000000000400fa7 <+23>:    add    $0x8,%rsp
   0x0000000000400fab <+27>:    retq
End of assembler dump.
(gdb)
```

2. 해당 명령어는 gdb에서 objdump와 같은 기능을 하는 명령어이다.

   1) disassemble [함수 명] : 함수의 어셈블리 코드를 출력한다.

   2) disassemble [주소 1] [주소 2] : 주소 1 ~ 주소 2 범위 사이의 어셈블리 코드를 출력한다.

Bomb Lab – Phase 1

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:
   0x0000000000400f90 <+0>:     sub    $0x8,%rsp
   0x0000000000400f94 <+4>:     mov    $0x402670,%esi
   0x0000000000400f99 <+9>:     callq  0x401398 <strings_not_equal>
   0x0000000000400f9e <+14>:    test   %eax,%eax
   0x0000000000400fa0 <+16>:    je     0x400fa7 <phase_1+23>
   0x0000000000400fa2 <+18>:    callq  0x401671 <explode_bomb>
   0x0000000000400fa7 <+23>:    add    $0x8,%rsp
   0x0000000000400fab <+27>:    retq
End of assembler dump.
(gdb)
```

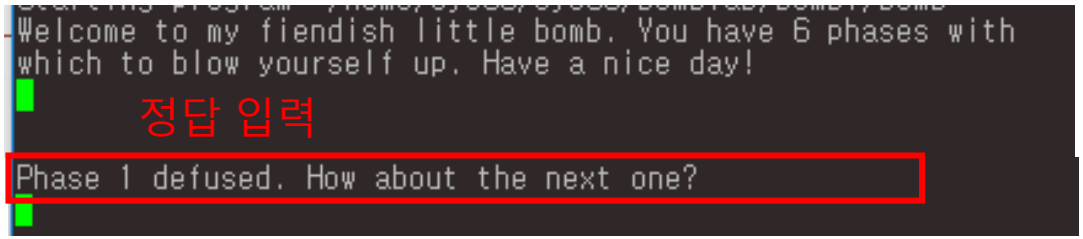
2.

3. 비교하는 값을 찾아야 하는데, 함수의 앞 부분에서 %esi의 값에 $0x402670 의 값을 옮기는 것을 볼 수 있다. 이 부분의 값을 보면 아래와 같다.

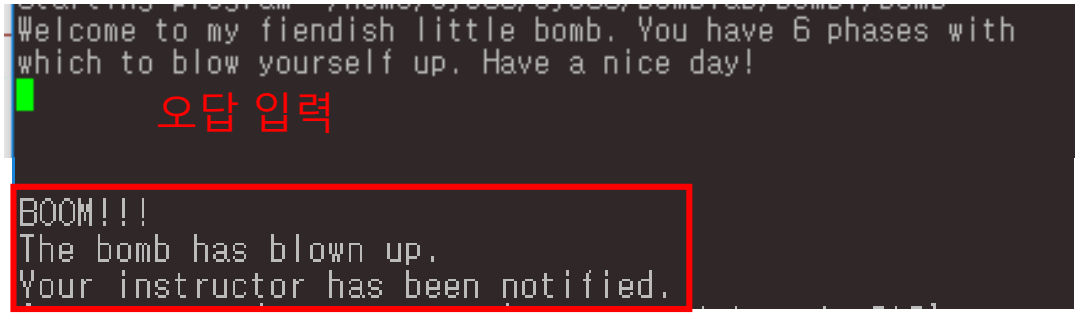
```
(gdb) x/s 0x402670
0x402670:       "We have to stand with our North Korean allies."
(gdb)
```

Bomb Lab – Phase 1

4. 앞에서 찾아낸 문자열을 입력하면 아래와 같이 다음 단계로 넘어간다.



5. 오답일 경우 아래와 같이 폭탄이 터지게 된다.



Bomb Lab – 풀이 방법

1. 결국 각 단계의 폭탄을 해체하기 위한 암호를 찾기 위해서는 어셈블리어로 변환된 코드의 구조를 이해해야 한다.

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

3. 이를 위하여 objdump로 전체 코드를 분석한 다음 gdb를 이용해서 하나 하나 추적해 나가는 과정이 필요하다.

Bomb Lab – 점수 계산

1. 아래의 웹 페이지를 통해 실시간으로 각자의 진행상황(해체/폭발)을 확인 할 수 있습니다.
   1) http:// 133.186.135.128:1500/scoreboard
   2) 폭탄을 모두 해체할 경우 60점
   3) 폭탄이 터진 경우
      - 2회 : -1점 / 20회 : -10점

2. Time Attack 방식으로 채점, 모든 폭탄(6 단계)을 제거했을 경우 완료 메일을 조교에게 보낸다.
   1) seula.hwang@cnu.ac.kr
   2) 메일 제목 : [sys00]폭탄완료_학번_이름

00: Bomb Lab Scoreboard

This page contains the latest information that we have received from your bomb. If your solution is marked **invalid**, this means your bomb reported a solution that didn't actually defuse your bomb.

Last updated: Mon Oct 31 15:18:50 2016 (updated every 30 secs)

#	Bomb number	Submission date	Phases defused	Explosions	Score	Status
1	bomb1	Mon Oct 31 15:10	1	1	10	valid

Summary [phase:cnt] [1:1] [2:0] [3:0] [4:0] [5:0] [6:0] total defused = 0/1

과제

1. Bomb Lab 기간

1) **11월 1일 ~ 11월 7일 (1주)**

2) 11월 7일 23시 59분 59초에 bomb lab 서버 종료

2. Bomb Lab 보고서

1) 결과 화면을 붙임(폭탄 해체 화면과 웹 페이지에서의 본인 결과)
2) 결과 화면에 학번이 보이도록 캡쳐
3) 풀이 과정에 대한 자세한 설명 (각 단계별 설명 + a)
4) Bomb Lab을 통해 느낀 점 및 에피소드, 하고 싶은 말
5) 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