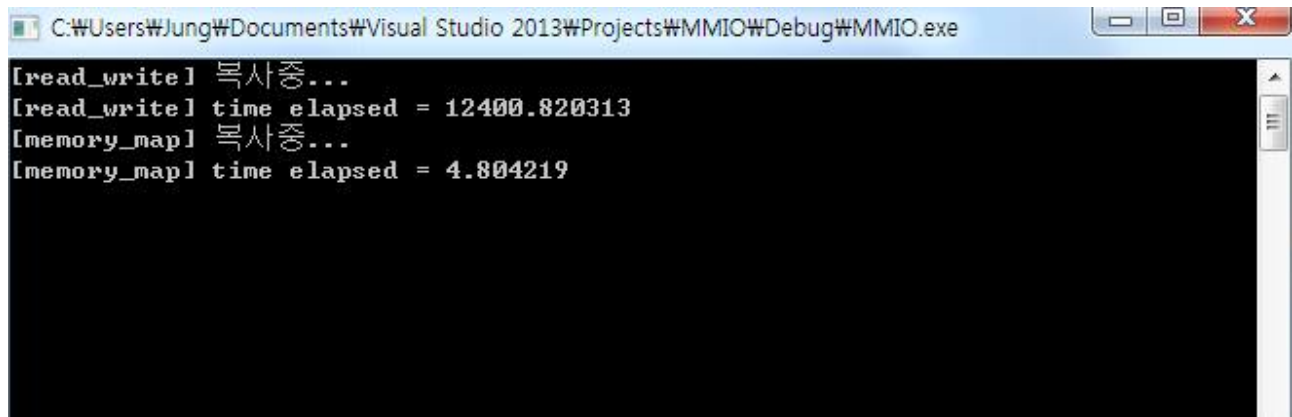


운영체제 과제 #2



제출일	2015. 07. 21	전공트랙	취약점분석
멘토	노용환	이름	정종빈

1. - [file_copy_using_memory_map] 과 [file_copy_using_read_write] 비교시간 출력



```
C:\Users\Jung\Documents\Visual Studio 2013\Projects\MMIO\Debug\MMIO.exe
[read_write] 복사중...
[read_write] time elapsed = 12400.820313
[memory_map] 복사중...
[memory_map] time elapsed = 4.804219
```

Memory_mapped 방식으로 하면 2480배 더 빠르게 읽고 쓸 수 있다.

2. - 파일 크기를 5G 이상하였을 때, 오류가 발생하는 원인과 해결 코드 작성

오류가 발생하는 원인은 메모리 저장공간이 부족해서 생기는 문제이다.

32비트 운영체제의 경우 가상메모리를 4기가를 이용하게 되는데, 2기가는 커널영역이 나머지 2기가를 유저영역에서 이용한다. 따라서 Memory mapped 방식의 I/O처리를 할 때 맵핑할 파일의 크기가 2기가가 넘어가면 커널영역의 공간을 침범하려고 하기 때문에 운영체제에서 보호하기 위해서 크래시를 발생시킨다. 해결 방법으로는 Memory mapped 크기를 2기가 이하로 나누어서 순차적으로 맵핑하고 쓰고 해제하고 다음 오프셋에서 다시 맵핑하고 쓰고 해제하고를 반복하면 간단하게 해결 할 수 있다.

즉, MapViewOfFile -> memcpy -> UnMapViewOfFile을 파일 전체를 모두 맵핑 할 때 까지 무한 반복하면 된다. 수정한 소스는 아래 그림을 참고하면 됩니다.

```
while (Size.QuadPart > 0){
    BytesToMap = Size.QuadPart > AllocationGranularity ? AllocationGranularity : Size.QuadPart;
    Size.QuadPart -= BytesToMap;
    src_ctx->view = (PCHAR)MapViewOfFile(
        src_ctx->map,
        FILE_MAP_ALL_ACCESS,
        AdjustOffset.HighPart,
        AdjustOffset.LowPart,
        BytesToMap
    );

    if (src_ctx->view == NULL)
    {
        //print("err ] MapViewOfFile( %ws ) failed. gle = %u", src_path, GetLastError());
        break;
    }

    dst_ctx->view = (PCHAR)MapViewOfFile(
        dst_ctx->map,
        FILE_MAP_WRITE | FILE_MAP_READ,
        AdjustOffset.HighPart,
        AdjustOffset.LowPart,
        BytesToMap
    );
    if (dst_ctx->view == NULL)
    {
        print("err ] MapViewOfFile( %ws ) failed. gle = %u", dst_path, GetLastError());
        break;
    }

    // copy src to dst by mmio
    for (uint32_t i = 0; i < BytesToMap ; ++i)
    {
        dst_ctx->view[i] = src_ctx->view[i];
    }

    UnmapViewOfFile(src_ctx->map);
    UnmapViewOfFile(dst_ctx->map);

    AdjustOffset.QuadPart += BytesToMap;
}
```