

## HOMEWORK2

취약점분석 임완호

### □ 파일생성

```
int _tmain(int argc, _TCHAR* argv[])
{
    printf("hello~ _tmain");

    _ASSERT(create_very_big_file(L"big.txt", 5000));

    Stopwatch sw;
    sw.Start();
    printf("file_copy_using_read_write\n");
    _ASSERT(file_copy_using_read_write(L"big.txt", L"big2.txt"));
    sw.Stop();
    printf("file_copy_using_read_write info] time elapsed = %f", sw.GetDurationSecond());

    Stopwatch sw2;
    sw2.Start();
    printf("file_copy_using_memory_map\n");
    _ASSERT(file_copy_using_memory_map(L"big.txt", L"big3.txt"));
    sw2.Stop();
    printf("file_copy_using_memory_map info] time elapsed = %f", sw2.GetDurationSecond());
    getchar();
    return 0;
}
```

- 5GB 이상의 파일을 생성후, read\_write 방식과 file\_copy\_using\_memory\_map 방식 두가지를 이용해 카피하는 시간을 잰다.

### □ Mapping 사이즈 지정

```
static DWORD AllocationGranularity = 0;
if (0 == AllocationGranularity)
{
    SYSTEM_INFO si = { 0 };
    GetSystemInfo(&si);
    AllocationGranularity = si.dwAllocationGranularity;
}

DWORD AdjustMask = AllocationGranularity - 1;
LARGE_INTEGER AdjustOffset = { 0 };
AdjustOffset.HighPart = Offset.HighPart;
// AllocationGranularity 이하의 값을 버림
AdjustOffset.LowPart = (Offset.LowPart & ~AdjustMask);
LONGLONG BytesToMap = (LONGLONG)((Offset.LowPart & AdjustMask) + (LONGLONG)(Size.QuadPart / 20));
// AllocationGranularity 이하의 값을 버림
// 버려진 값만큼 매핑할 사이즈를 증가
```

- 한번에 4G 이상을 맵핑시키면 메모리 오류가 발생하므로 입력한 Size를 20으로 나눈만큼씩 차례대로 맵핑 시킨다.

## □ 메모리 Mapping

```
while (dst_ctx->size.QuadPart > 0){  
    // 버려진 값만큼 매핑할 사이즈를 증가  
    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;  
    }  
}
```

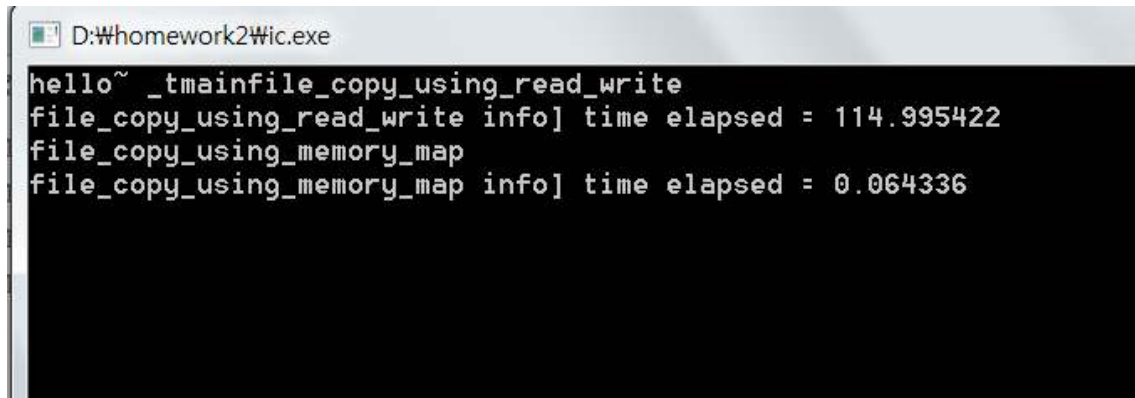
- 나눈 맵핑 사이즈만큼 src 파일과 dst파일을 메모리에 맵핑시킨다.

## □ 복사

```
// copy src to dst by mmio  
for (uint32_t i = 0; i < BytesToMap + AdjustOffset.LowPart; ++i)  
{  
    dst_ctx->view[i] = src_ctx->view[i];  
}  
  
UnmapViewOfFile(src_ctx->map);  
UnmapViewOfFile(dst_ctx->map);  
  
AdjustOffset.LowPart += BytesToMap;  
dst_ctx->size.QuadPart -= BytesToMap;  
}
```

- 소스에서 목적지로 mmio를 수행한다.
- Offset을 ByteToMap 만큼 증가시켜 다음번에 mmio를 시행할 부분으로 옮겨준다
- 파일크기를 맵핑한 만큼 감소시켜 메모리매핑 while 조건문이 충족할동안 반복한다.

## □ 수행시간비교



```
hello~ _tmainfile_copy_using_read_write
file_copy_using_read_write info] time elapsed = 114.995422
file_copy_using_memory_map
file_copy_using_memory_map info] time elapsed = 0.064336
```

- read\_write 수행시간이 memory\_map 수행시간보다 훨씬 오래걸리는 것을 알 수 있다.  
(약 1787배 오래걸림)

## □ 결과

이름	수정한 날짜	유형	크기
big.txt	2015-07-21 오전 1...	텍스트 문서	5,120,000KB
big2.txt	2015-07-21 오전 1...	텍스트 문서	5,120,000KB
big3.txt	2015-07-21 오전 1...	텍스트 문서	5,120,000KB