과제 2 - 2018131321 김현우

✏️ CVE-2017-5225(LibTIFF)

```
# 필요한 패키지 설치
sudo apt update
sudo apt install build-essential wget libjpeg-dev zlib1g-dev

# 소스 디렉토리로 이동 및 다운로드
wget http://download.osgeo.org/libtiff/tiff-4.0.7.tar.gz
tar xzf tiff-4.0.7.tar.gz
cd tiff-4.0.7

# configure 스크립트 실행
./configure

# 빌드
make
```

- Ubuntu 20.04 환경에서 다음과 같은 명령어를 실행해 4.0.7버전의 LibTIFF를 다운받는다
- 다운 완료 후 빌드를 진행한다.

- poc.tiff 파일을 `tools` 패키지에 넣는다

✏️ Trigger the vulnerability

- PoC 두 가지 경우가 있으므로, 각각의 PoC파일을 필요할 때 넣고 vulnerability를 유발시킨다.

✔️ trigger 1

```
w009981@LAPTOP-QC3DDJDH:~/tiff-4.0.7/tools$ ./tiffcp -p contig poc.tiff output.tiff
TIFFReadDirectoryCheckOrder: Warning, Invalid TIFF directory; tags are not sorted in ascending order.
TIFFReadDirectory: Warning, Unknown field with tag 233 (0xe9) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 768 (0x300) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 3 (0x3) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 26996 (0x6974) encountered.
TIFFFetchNormalTag: Warning, IO error during reading of "DocumentName"; tag ignored.
TIFFFetchNormalTag: Warning, Incorrect count for "Orientation"; tag ignored.
TIFFFetchNormalTag: Warning, Incorrect count for "XResolution"; tag ignored.
Fax3Decode2D: Bad code word at line 0 of strip 0 (x 256).
Fax3Decode2D: Warning, Premature EOL at line 0 of strip 0 (got 256, expected 285).
malloc(): mismatching next->prev_size (unsorted)
Aborted
```

- `cpSeparate2ContigByRow` 함수에서 힙 버퍼 오버플로우가 발생한다
- for loop `imagewidth` 변수가 `scanlinesizeout`보다 클 때 오류가 발생한다.
- `imagewidth` 가 `scanlinesizeout` 보다 크면, 루프가 지정된 경계를 넘게 된다.

✔️ trigger 2

```
w009981@LAPTOP-QC3DDJDH:~/tiff-4.0.7/tools$ ./tiffcp -p separate poc.tiff output.tiff
corrupted size vs. prev_size
Aborted
```

- `cpContig2SeparateByRow` 함수에서 힙 버퍼 오버플로우가 발생한다.
- for 루프에서 `imagewidth` 변수가 `scanlinesizeout` 또는 `scanlinesizein/spp` (samples per pixel)보다 클 때 오류가 발생한다.
- out of bound write or read가 발생할 수 있다.

✏️ Apply security patch

tiffcp.c

```
static int
tiffcp(TIFF* in, TIFF* out)
{
        uint16 bitspersample = 1, samplesperpixel = 1;
        uint16 input_compression, input_photometric = PHOTOMETRIC_MINISBLACK;
        copyFunc cf;
        uint32 width, length;
```

```
        register uint32 n;
        uint32 row;
        tsample_t s;
        uint16 bps = 0;

        (void) TIFFGetField(in, TIFFTAG_BITSPERSAMPLE, &bps);
        if( bps != 8 )
        {
            TIFFError(TIFFFileName(in),
                      "Error, can only handle BitsPerSample=8 in %s",
                      "cpContig2SeparateByRow");
            return 0;
        }

        inbuf = _TIFFmalloc(scanlinesizein);
        outbuf = _TIFFmalloc(scanlinesizeout);
```

```
        uint32 row;
        tsample_t s;
        uint16 bps = 0;

        (void) TIFFGetField(in, TIFFTAG_BITSPERSAMPLE, &bps);
        if( bps != 8 )
        {
            TIFFError(TIFFFileName(in),
                      "Error, can only handle BitsPerSample=8 in %s",
                      "cpSeparate2ContigByRow");
            return 0;
        }

        inbuf = _TIFFmalloc(scanlinesizein);
        outbuf = _TIFFmalloc(scanlinesizeout);
```

```
static copyFunc
pickCopyFunc(TIFF* in, TIFF* out, uint16 bitspersample, uint16 samplesperpixel)
{
        uint16 shortv;
        uint32 w, l, tw, tl;
        int bychunk;

        (void) TIFFGetFieldDefaulted(in, TIFFTAG_PLANARCONFIG, &shortv);
        if (shortv != config && bitspersample != 8 && samplesperpixel > 1) {
                fprintf(stderr,
                    "%s: Cannot handle different planar configuration w/ bits/sample != 8\n",
                    TIFFFileName(in));
                return (NULL);
        }
```

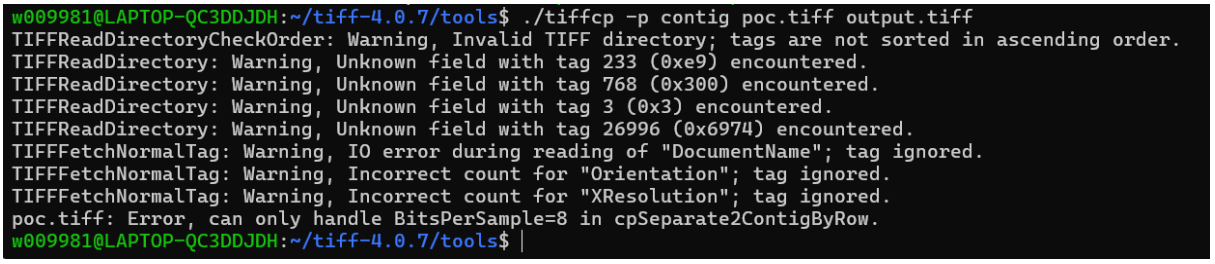
✏️ Ensure the vulnerability is safely remediated

Rebuild

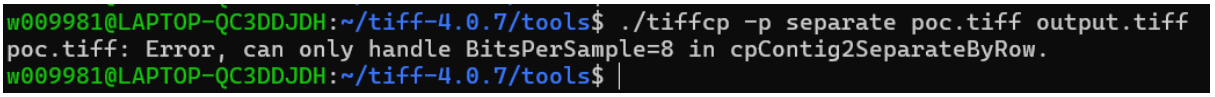
```
make clean

# configure 스크립트 실행
./configure

# 빌드
make
```

- `tiffcp.c` 파일을 수정한 후, 다시 rebuild를 진행한다.

✔️ patch 1



✔️ patch 2



- patch를 적용하면, `BitsPerSample` 을 확인해 8이 아닌 경우 오류를 발생 시킨다.
- 8인 경우 나머지 코드를 실행해 메모리 접근이 올바르게 이루어지도록 보장한다.

- TIFF의 경우 일반적으로 8, 16 비트 등으로 설정되는데, bps가 8이 아닌 경우 메모리 접근 인덱스 계산이 딱 맞지 않는다.
- 즉, 메모리 할당 등이 복잡해져 heap buffer overflow가 발생할 가능성이 높아진다.
- 패치 코드는 이를 방지하기 위해 `BitsPerSample` 값이 8이 아닌 경우 함수 실행을 중단하여 안전한 메모리 접근을 보장한다.