本文作者： 阿尔法实验室漏洞报送 | 2019年4月3日 | CVE |

天融信关于LibTiff中invertImage函数堆溢出漏洞的分析

一、背景介绍

LibTIFF 是一个用来读写标签图像文件格式（简写为TIFF）的库。这个库还包含一些命令行工具用来处理TIFF文件。它以源代码方式分发，并且可以在多种平台上以二进制构建的方式出现。LibTIFF软件由Sam Leffler在Silicon Graphics工作期间编写。

1.1 漏洞描述

在LibTIFF4.0.10版本中，tiffcrop工具的invertImage()函数没有正确处理色彩深度为2和4的情况，导致堆缓冲区溢出。

1.2 受影响的系统版本

LibTIFF4.0.10

二、环境搭建

1.下载源码

https://download.osgeo.org/libtiff/tiff-4.0.10.zip

2.按默认选项编译

三、漏洞分析

tiff-4.0.10/tools/tiffcrop.c文件中，invertImage()函数的代码有几个问题，该函数用于反转bilevel或grayscale类型TIFF图像的明暗值。在第9206行，可以看到一个switch语句用于处理各种色彩深度的情况。

```
9142  invertImage(uint16 photometric, uint16 spp, uint16 bps, uint32 width, uint32 length, unsigned char *work_buff)
9143    {
9144    uint32   row, col;
9145    unsigned char  bytebuff1, bytebuff2, bytebuff3, bytebuff4;
9146    unsigned char *src;
9147    uint16        *src_uint16;
9148    uint32        *src_uint32;
9149
9150    if (spp != 1)
9156    if (photometric !=  PHOTOMETRIC_MINISWHITE && photometric !=  PHOTOMETRIC_MINISBLACK)
9162    src = work_buff;
9163    if (src == NULL)
9169    switch (bps)
9170      {
9171      case 32: src_uint32 = (uint32 *)src;
9179      case 16: src_uint16 = (uint16 *)src;
9187      case 8: for (row = 0; row < length; row++)
9190      *src = (uint8)255 - *src;
9194      case 4: for (row = 0; row < length; row++)
9197      bytebuff1 = 16 - (uint8)(*src & 240 >> 4);
9198      bytebuff2 = 16 - (*src & 15);
9199      *src = bytebuff1 << 4 & bytebuff2;
9203      case 2: for (row = 0; row < length; row++)
9206      bytebuff1 = 4 - (uint8)(*src & 192 >> 6);
9207      bytebuff2 = 4 - (uint8)(*src & 48  >> 4);
9208      bytebuff3 = 4 - (uint8)(*src & 12  >> 2);
9209      bytebuff4 = 4 - (uint8)(*src & 3);
9210      *src = (bytebuff1 << 6) | (bytebuff2 << 4) | (bytebuff3 << 2) | bytebuff4;
9214      case 1: for (row = 0; row < length; row++)
9221      default: TIFFError("invertImage", "Unsupported bit depth %d", bps);
9223      }
```

但注意在case 2语句中，由于某些未知原因，在处理bps值2和4的情况时，反转比特位的方法和32,16,8和1不同。这里循环迭代处理4个像素，同时还进行了宽度迭代，最后在移位操作时导致了堆缓冲区溢出。

```
    case 2: for (row = 0; row < length; row++)
              for (col = 0; col < width; col++)
                {
                  bytebuff1 = 4 - (uint8)(*src & 192 >> 6);
                  bytebuff2 = 4 - (uint8)(*src & 48  >> 4);
                  bytebuff3 = 4 - (uint8)(*src & 12  >> 2);
                  bytebuff4 = 4 - (uint8)(*src & 3);
                  *src = (bytebuff1 << 6) | (bytebuff2 << 4) | (bytebuff3 << 2) | bytebuff4;
                  src++;
                }
             break;
```

```
  case 2: for (row = 0; row < length; row++)
            for (col = 0; col < width; col++)
              {
                bytebuff1 = 4 - (uint8)(*src & 192 >> 6);
                bytebuff2 = 4 - (uint8)(*src & 48  >> 4);
                bytebuff3 = 4 - (uint8)(*src & 12  >> 2);
                bytebuff4 = 4 - (uint8)(*src & 3);
                *src = (bytebuff1 << 6) | (bytebuff2 << 4) | (bytebuff3 << 2) | bytebuff4;
                src++;
              }
           break;
```

所以在这里，将所有位反转使用255-x =~x这样的方法更容易，下面是修复方案。

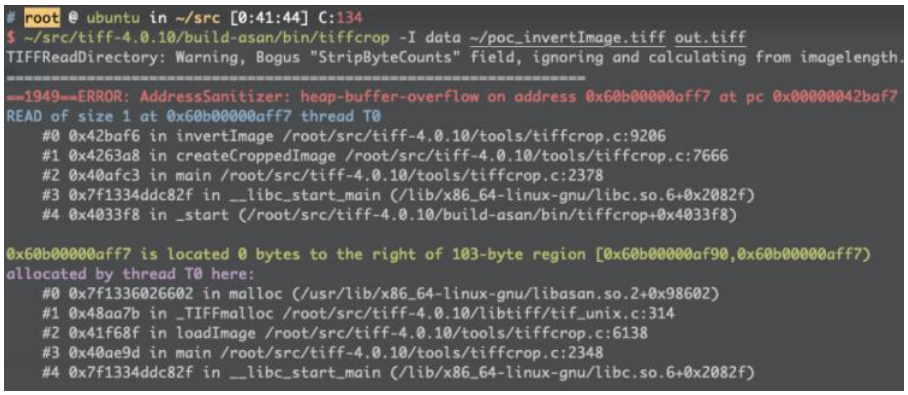
tools/tiffcrop.c — View file @ 9cfa5c46

```
9176 9175                 {
9177      -               *src_uint32 = (uint32)0xFFFFFFFF - *src_uint32;
     9176 +               *src_uint32 = ~(*src_uint32);
9178 9177                  src_uint32++;
9179 9178                 }
9180 9179             break;
 ...  ...  @@ -9182,39 +9181,15 @@ invertImage(uint16 photometric, uint16 spp, uint16 bps, uint32 width, uint32 len
9182 9181              for (row = 0; row < length; row++)
9183 9182                for (col = 0; col < width; col++)
9184 9183                 {
9185      -               *src_uint16 = (uint16)0xFFFF - *src_uint16;
     9184 +               *src_uint16 = ~(*src_uint16);
9186 9185                  src_uint16++;
9187 9186                 }
9188 9187             break;
9189      -      case 8: for (row = 0; row < length; row++)
9190      -                for (col = 0; col < width; col++)
9191      -                  {
9192      -                  *src = (uint8)255 - *src;
9193      -                   src++;
9194      -                  }
9195      -             break;
9196      -      case 4: for (row = 0; row < length; row++)
9197      -                for (col = 0; col < width; col++)
9198      -                  {
9199      -                  bytebuff1 = 16 - (uint8)(*src & 240 >> 4);
9200      -                  bytebuff2 = 16 - (*src & 15);
9201      -                  *src = bytebuff1 << 4 & bytebuff2;
9202      -                  src++;
9203      -                  }
9204      -             break;
9205      -      case 2: for (row = 0; row < length; row++)
9206      -                for (col = 0; col < width; col++)
9207      -                  {
9208      -                  bytebuff1 = 4 - (uint8)(*src & 192 >> 6);
9209      -                  bytebuff2 = 4 - (uint8)(*src & 48  >> 4);
9210      -                  bytebuff3 = 4 - (uint8)(*src & 12  >> 2);
9211      -                  bytebuff4 = 4 - (uint8)(*src & 3);
9212      -                  *src = (bytebuff1 << 6) | (bytebuff2 << 4) | (bytebuff3 << 2) | bytebuff4;
9213      -                  src++;
9214      -                  }
9215      -             break;
     9188 +      case 8:
     9189 +      case 4:
     9190 +      case 2:
9216 9191           case 1: for (row = 0; row < length; row++)
9217      -                for (col = 0; col < width; col += 8 /(spp * bps))
     9192 +                for (col = 0; col < width; col += 8 / bps)
9218 9193                   {
9219 9194                   *src = ~(*src);
9220 9195                   src++;
```

四、漏洞利用

使用libtiff中的tiffcrop工具反转图像色彩空间，导致DoS：

tiffcrop -I data poc.tiff out.tiff



Written by 阿尔法实验室漏洞报送

Read other posts by 阿尔法实验室漏洞报送

← Previous
天融信关于LibTiff中PS_Lvl2page函数堆溢出漏洞的分析

Next →
天融信关于LayerBB 1.1.3 xss漏洞分析

阿尔法实验室
ALPHA LAB OF TOPSEC