

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（简称为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) -
9151 # if (photometric != PHOTOMETRIC_MINISWHITE && photometric != PHOTOMETRIC_MINISBLACK) -
9152     src = work_buff;
9153 # if (src == NULL) -
9154     switch (bps)
9155     {
9156 # case 32: src_uint32 = (uint32 *)src; -
9157 # case 16: src_uint16 = (uint16 *)src; -
9158 # case 8: for (row = 0; row < length; row++) -
9159 # *src = (uint8)255 - *src; -
9160 # case 4: for (row = 0; row < length; row++) -
9161 # bytebuff1 = 16 - (uint8)(*src & 240 >> 4);
9162 # bytebuff2 = 16 - (*src & 15);
9163 # *src = bytebuff1 << 4 & bytebuff2; -
9164 # case 2: for (row = 0; row < length; row++) -
9165 # bytebuff1 = 4 - (uint8)(*src & 192 >> 6);
9166 # bytebuff2 = 4 - (uint8)(*src & 48 >> 4);
9167 # bytebuff3 = 4 - (uint8)(*src & 12 >> 2);
9168 # bytebuff4 = 4 - (uint8)(*src & 3);
9169 # *src = (bytebuff1 << 6) | (bytebuff2 << 4) | (bytebuff3 << 2) | bytebuff4; -
9170 # case 1: for (row = 0; row < length; row++) -
9171 # default: TIFFError("invertImage", "Unsupported bit depth %d", bps); -
9172 }
```

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。

文章归档

2022年五月
2022年四月
2022年一月
2021年十二月
2021年十一月
2021年十月
2021年九月
2021年七月
2021年四月
2021年三月
2021年一月
2020年十二月
2020年十月
2020年九月
2020年八月
2020年七月
2020年六月
2020年五月
2020年四月
2020年三月
2020年一月
2019年十二月
2019年十一月
2019年十月
2019年九月
2019年八月
2019年七月
2019年六月
2019年五月
2019年四月
2019年三月
2019年二月
2019年一月
2018年十一月
2018年十月
2018年九月
2018年八月
2018年七月
2018年六月
2018年五月
2018年四月
2018年三月
2018年一月
2017年十二月
2017年十月
2017年九月
2017年八月
2017年七月
2017年六月
2017年五月
2017年四月
2017年二月
2016年十二月
2016年十月
2016年九月
2016年八月
2016年六月
2016年五月
2016年四月
2016年三月
2016年二月
2016年一月
2015年十二月
2015年十一月
2015年十月
2015年九月

2015年七月
2015年六月
2015年五月
2015年二月
2014年十二月
2014年十月
2014年七月
2014年六月
2014年五月
2014年四月

```
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		for (col = 0; col < width; col++)
9177	-	*src_uint32 = (uint32)0xFFFFFFFF - *src_uint32;	{
9176	+	*src_uint32 = ~(*src_uint32);	src_uint32++;
9178	9177		}
9179	9178		break;
9180	9179		...
...	...	@@ -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;	src_uint16++;
9184	+	*src_uint16 = ~(*src_uint16);	}
9186	9185		break;
9187	9186		case 8: for (row = 0; row < length; row++)
9188	9187		for (col = 0; col < width; col++)
9189	-		{
9190	-		*src = (uint8)255 - *src;
9191	-		src++;
9192	-		}
9193	-		break;
9194	-		case 4: for (row = 0; row < length; row++)
9195	-		for (col = 0; col < width; col++)
9196	-		{
9197	-		bytebuff1 = 16 - (uint8)(*src & 240 >> 4);
9198	-		bytebuff2 = 16 - (*src & 15);
9199	-		*src = bytebuff1 << 4 & bytebuff2;
9200	-		src++;
9201	-		}
9202	-		break;
9203	-		case 2: for (row = 0; row < length; row++)
9204	-		for (col = 0; col < width; col++)
9205	-		{
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;
9211	-		src++;
9212	-		}
9213	-		break;
9214	-		case 1: for (row = 0; row < length; row++)
9215	-		for (col = 0; col < width; col += 8 / (spp * bps))
9188	+	case 8:	
9189	+	case 4:	
9190	+	case 2:	
9216	9191		for (col = 0; col < width; col += 8 / bps)
9217	-		{
9218	9193		*src = ~(*src);
9219	9194		src++;
9220	9195		

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
```

```
# root @ ubuntu in ~/src [0:41:44] C:134
$ ~/src/tiff-4.0.10/build-asan/bin/tiffcrop -I data ~/poc_invertImage.tiff out.tiff
TIFFReadDirectory: Warning, Bogus "StripByteCounts" field, ignoring and calculating from imagelength.
=====
--1949--ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60b00000aff7 at pc 0x00000042baf7
READ of size 1 at 0x60b00000aff7 thread T0
#0 0x42baf6 in invertImage /root/src/tiff-4.0.10/tools/tiffcrop.c:9206
#1 0x4263a8 in createCroppedImage /root/src/tiff-4.0.10/tools/tiffcrop.c:7666
#2 0x40afc3 in main /root/src/tiff-4.0.10/tools/tiffcrop.c:2378
#3 0x7f1334ddc82f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
#4 0x4033f8 in _start (/root/src/tiff-4.0.10/build-asan/bin/tiffcrop+0x4033f8)

0x60b00000aff7 is located 0 bytes to the right of 103-byte region [0x60b00000af90,0x60b00000aff7)
allocated by thread T0 here:
#0 0x7f1336026602 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
#1 0x48aa7b in _TIFFmalloc /root/src/tiff-4.0.10/libtiff/tif_unix.c:314
#2 0x41f68f in loadImage /root/src/tiff-4.0.10/tools/tiffcrop.c:6138
#3 0x40ae9d in main /root/src/tiff-4.0.10/tools/tiffcrop.c:2348
#4 0x7f1334ddc82f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
```

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](#)



Copyright 2022 天融信阿尔法实验室