FFmpeg任意文件读取漏洞分析

阿里聚安全 / 2017-08-02 06:31:39 / 浏览数 3465 安全技术 漏洞分析 顶(0) 踩(0)

6月24号的时候hackerone网站上公布了一个ffmpeg的本地文件泄露的漏洞，可以影响ffmpeg很多版本，包括3.2.2、3.2.5、3.1.2、2.6.8等等。

hackerone网站上的漏洞介绍:
https://hackerone.com/reports/226756。与之相关联的还有其他几篇hackerone上的漏洞报告：https://hackerone.com/reports/243470、https://hackerone.com/re

实际上这个漏洞@EmilLerner和@PavelCheremushkin最早在今年的phdays大会上就已经披露了，视频可以在https://www.phdays.com/broadcast/搜索Attacks on video converter得到，不过是全俄语的，暂时没有英文字母。也可以参考漏洞作者在phday大会上的ppt

作者实际上在5月份的时候就发现了该漏洞并提交，6月26号的时候公布出来，最新的ffmpeg版本3.3.2中已经修复了，但是旧的版本不保证。

背景介绍
FFmpeg是一套目前非常流行的可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。它提供了录制、转换以及流化音视频的完整解决方案。目前有非
APP 都采用了这个库，但是这个库历史上曝出的漏洞也非常之多。这次的漏洞是利用了ffmpeg可以处理 HLS 播放列表的功能，在 AVI 文件中的
GAB2字幕块中嵌入了一个 HLS 文件，然后提供给ffmpeg进行转码，在解析的过程中把它当做一个 XBIN 的视频流来处理，再通过 XBIN
的编解码器把本地的文件包含进来，最后放在转码后的视频文件当中。

漏洞重现方法
1) 下载脚本 https://github.com/neex/ffmpeg-avi-m3u-xbin/blob/master/gen_xbin_avi.py2) 运行脚本：python3 gen_xbin_avi.py file:///etc/passwd
sxcurity.avi3) 访问https://arxius.io，上传sxcurity.avi4) 等待上传处理视频5) 录像处理完成后，点击播放就可以看到/etc/passwd的内容

温故而知新
这次的漏洞实际上与之前曝出的一个 CVE 非常之类似，可以说是旧瓶装新酒，老树开新花。

之前漏洞的一篇分析文章：
SSRF 和本地文件泄露（CVE-2016-1897/8）http://static.hx99.net/static/drops/papers-15598.html

这个漏洞实际上也是利用了ffmpeg在处理 HLS 播放列表文件的过程中，由于支持非常多的协议，如http、file、concat等等，导致可以构造恶意的url造成 SSRF
攻击和本地文件泄露。下面这幅图介绍了整个的攻击流程。

官方对这个漏洞的修复是把concat协议加入了黑名单并在结构体中加入了protocol_blacklist和protocol_whitelist这两个域。但是这次的漏洞利用了内嵌在字幕块中的
m3u8文件成功绕过了ffmpeg的某些限制。

HLS 协议简单介绍
HLS(HTTP Live Streaming)是苹果公司针对iPhone、iPod、iTouch和iPad等移动设备而开发的基于HTTP协议的流媒体解决方案。在 HLS 技术中 Web
服务器向客户端提供接近实时的音视频流。但在使用的过程中是使用的标准的 HTTP 协议，所以这时，只要使用 HLS 的技术，就能在普通的 HTTP
的应用上直接提供点播和直播。该技术基本原理是将视频文件或视频流切分成小片(ts)并建立索引文件(m3u8)。客户端会先向服务器请求
m3u8索引文件，然后根据索引文件里面的url去请求真正的ts视频文件。如果是多级的m3u8索引的话，那就会从根索引文件开始，一层一层的往下去请求子的索引文件，获

M3U8文件中有很多TAG，每一个 TAG 的详细的作用可以参考这篇文章：http://blog.csdn.net/cabbage2008/article/details/50522190

针对ffmpeg 3.1.2版本的漏洞分析
先介绍一下ffmpeg是怎样把一个输入文件转码成另一种格式的视频文件，流程图如下：（图片引用自http://blog.csdn.net/leixiaohua1020/article/details/25422685）

然后我们观察一下攻击者提供的poc生成的sxcurity.avi文件的结构

最开始是 AVI 文件的文件头，然后中间有一个 GAB2字幕的文件头"GAB2"，在 GAB2的文件头后面还有 HLS 播放列表的文件头"#EXTM3U"，在文件头后面就是
HLS 的文件内容了。

这次的漏洞主要是利用了ffmpeg处理 GAB2字幕块的时候的逻辑错误，所以我们重点从ffmpeg打开输入文件时调用
read_gab2sub()这个函数开始分析，read_gab2sub() 函数是被avformat_find_stream_info() 这个函数调用的。整个函数调用流程图如下：

我们可以看到在为播放列表确定demuxer的时候探测出格式为XBIN，但是单看sxcurity.avi文件中并没有 XBIN 的文件头，那么ffmpeg是怎么确定格式是 XBIN
的呢。这就与 HLS 的解析过程密切相关了。在判断一个播放列表的格式时候，ffmpeg会读取播放列表的第一个segment，然后根据这个 segment
的url去请求文件内容，然后根据读取到的文件内容来判断格式。
我们可以看到sxcurity.avi里面内嵌的 playlist 的前几行是这样写的#EXT-X-MEDIA-SEQUENCE:0

echoing b'XBIN\x1a \x00\x0f\x00\x10\x04\x01\x00\x00\x00\x00'

EXT-X-KEY: METHOD=AES-128, URI=/dev/zero, IV=0x4c4d465e0b95223279487316ffd9ec3a

EXTINF:1,

EXT-X-BYTERANGE: 16

/dev/zero

EXT-X-KEY: METHOD=NONE

echoing b'\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\xbf\n'

EXT-X-KEY: METHOD=AES-128, URI=/dev/zero, IV=0x140f0f1011b5223d79597717ffd95330

以#EXT开头的都是m3u8文件中的标签，其他以#开头的是注释，不以#开头的是一个url

介绍一下上面出现的标签的含义

EXT-X-MEDIA-SEQUENCE:每一个media URI 在 PlayList中只有唯一的序号，相邻之间序号+1, 一个media URI并不是必须要包含的，如果没有，默认为0。

EXTINF: duration 指定每个媒体段(ts)的持续时间（秒），仅对其后面的URI有效。

EXT-X-KEY表示怎么对mediasegments进行解码。其作用范围是下次该tag出现前的所有mediaURI，属性为NONE或者AES-128。对于AES-128的情况，URI属性表示一个key文件，通过URI可以获得这个key，如果没有IV（Initialization Vector）,则使用序列号作为IV进行编解码；如果有IV，则将改值当成16个字节的16进制数。

那么上面几行代码包括了两个segment，每一个 segment 都需要使用 AES-128进行解密，密文是从/dev/zero这个 url获取的16个字节，其实就是16个'\0'， key 也是从/dev/zero这个url获取的16个字节，16个'\0'，IV是自己指定的一串十六进制数。ffmpeg对第一个 segment 进行解密后的结果是b'XBIN\x1a \x00\x0f\x00\x10\x04\x01\x00\x00\x00\x00', 和漏洞作者在注释中给出的一样。前四个字节 XBIN 就是 XBIN 的文件头。从而使得ffmpeg判定这个播放列表的格式是 XBIN。

在open_input_file的函数结尾，程序会调用av_dump_format函数打印格式信息，打印结果如下
Input #0, avi, from '../sxcurity.avi': Duration: 00:00:05.00, start: 0.000000, bitrate: 547 kb/s   Stream #0:0: Video: xbin, pal8, 256x240, 25 tbr, 25 tbn, 25 tbc

可以看到文件中只有一个视频流，视频流的格式编码格式被识别成了xbin
在打开输入文件和打开输出文件之后，程序就要开始进行转码工作了
在进行转码的过程中，我们可以在get_input_packet函数上下断点，该函数的作用是获取输入的一帧压缩的视频数据放在AVPacket中。

gdb-peda$ bt#0 get_input_packet (f=0x6dfc00, pkt=0x7fffffffe460) at ffmpeg.c:3663

1 0x000000000042d40c in process_input (file_index=0x0) at ffmpeg.c:3798

2 0x000000000042eed2 in transcode_step () at ffmpeg.c:4108

3 0x000000000042effc in transcode () at ffmpeg.c:4162

4 0x000000000042f716 in main (argc=0x4, argv=0x7fffffffea48) at ffmpeg.c:4357

5 0x00007ffff48a543a in __libc_start_main () from /usr/lib/libc.so.6

6 0x0000000000407ada in _start ()

get_input_packet() 调用结束之后打印pkt的内容，可以看到里面已经有了/etc/passwd文件的开头的部分信息。

然后 ffmpeg会转码成mpeg2的编码格式，最后封装成一个 mp4文件。打开 MP4文件，我们就能看到泄露出的文件内容了。

不转码的avi文件也是可以播放的，只是他播放的文件是当前调用ffmpeg的应用，例如你用爱奇艺打开就是调用爱奇艺的播放器，而爱奇艺是用了ffmpeg代码，就可以读到

例如可以读到data/data/com.qiyi.video/shared_prefs/cn.com.mma.mobile.tracking.sdkconfig.xml

而转码的目的就是把你上传视频的服务器文件写死在转码后的文件，就是你无论转成.mp4,flv都可以播放的。

另一种攻击方式
攻击者针对https://hackerone.com/reports/242831这个洞的 fix 又提出了另一种 bypass 方式https://hackerone.com/reports/243470。这个方式比起使用xbin的方法要来的更加简洁。

这次的 POC 生成的playlist长下面这样

EXTM3U

EXT-X-MEDIA-SEQUENCE:0

EXTINF:1.0

GOD.txt

EXTINF:1.0

/etc/passwd

EXT-X-ENDLIST

翻译一下作者在评论区的说明：
首先我们有几点需要了解关于HLS playlist 是怎么被处理的：
1.处理一个 playlist 的时候ffmpeg把所有 segment 的内容连接在一起然后当做一个单独的文件来处理
2.ffmpeg会使用 playlist 第一个 segment 来决定文件的类型
3.ffmpeg用一种特殊的方式来处理.txt后缀的文件，它会尝试将文件的内容以终端的方式打印在屏幕上

所以上面的 playlist 的处理流程是这样的：
1.ffmpeg在 GAB2字幕块里面看到了#EXTM3U标签，认定文件类型是 HLS playlist。
2.GOD.txt这个文件甚至不需要存在，但是它的名字足够ffmpeg把文件类型检测成txt类型了
3.ffmpeg把 playlist 的所有 segment 的内容连接在一起，因为只有/etc/passwd这个文件是实际存在的，所以最后的内容就是/etc/passwd文件的内容
4.因为这个文件的类型是 txt 类型，所以ffmpeg绘制一个终端来打印这个文件。

需要注意的是在解析 playlist 文件的时候，每一个 segment 的url协议d的白名单为'file,
crypto'，所以这里直接把/etc/passwd改成http://vul.com:21进行ssrf攻击是行不通的。

对官方补丁的分析
补丁链接：https://github.com/FFmpeg/FFmpeg/commit/189ff4219644532bdfa7bab28dfedaee4d6d4021
官方对这个漏洞的修复也很简单，只是对播放列表中 file
协议的文件扩展名使用了白名单进行过滤。一定程度上缓解了攻击，但是还是可以泄露出那些多媒体文件。

```
typedefstructHLSContext {
        //...省略了一下结构体的内容
+    char allowed_extensions; //增添一个字段，限制了允许的文件扩展名
} HLSContext;
staticintopen_url(AVFormatContext s,
AVIOContext *pb, constchar url,AVDictionary opts, AVDictionary opts2, int *is_http)
{
        //...
// only http(s) & file are allowed
-    if (!av_strstart(proto_name, "http", NULL) && !av_strstart(proto_name, "file", NULL))
+    if (av_strstart(proto_name, "file", NULL)) {
+        if (strcmp(c->allowed_extensions, "ALL") && !av_match_ext(url, c->allowed_extensions)) {
+            av_log(s, AV_LOG_ERROR,
+                "Filename extension of \'%s\' is not a common multimedia extension, blocked for security reasons.\n"
+                "If you wish to override this adjust allowed_extensions, you can set it to \'ALL\' to allow all\n",
+                url);
+            return AVERROR_INVALIDDATA;
+        }
+    } elseif (av_strstart(proto_name, "http", NULL)) {
+        ;
+    } elsereturn AVERROR_INVALIDDATA;
+
//...
}
```

总结
这个漏洞和以往的核心是m3u8文件可以根据指定url获取图片文字，而它里面的http协议和file协议没有过滤好，导致可以ssrf和读取任意文件，以前的漏洞是利用concat可

参考资料
http://blogs.360.cn/blog/ffmpegs_security_discussion/
https://github.com/radman1/xbin
https://hackerone.com/reports/243470
http://static.hx99.net/static/drops/papers-15598.html

https://tools.ietf.org/html/draft-pantos-http-live-streaming-18#userconsent
http://blog.csdn.net/cabbage2008/article/details/50522190
http://blogs.360.cn/blog/ffmpegs_security_discussion/
XBIN 文件格式的详细介绍：https://en.wikipedia.org/wiki/XBin

- 作者：栈长、胖苗、超六@蚂蚁金服，更多安全知识分享和热点信息，请关注阿里聚安全的官方博客

上一篇：甲方工作杂谈 下一篇：0729JSRC沙龙PPT-兜哥

1. 0 条回复
    - 动动手指，沙发就是你的了！