

FFmpeg 漏洞挖掘之路

谢俊东（栈长）

蚂蚁金服巴斯光年安全攻防团队



什么是 Fuzz （模糊测试）

- 一种软件测试技术
- 自动或半自动的生成随机的测试数据
- 监视程序异常(crash,oom,timeout等等)



传统 Fuzz的特点

- 随机构造数据
- 对于大型、复杂程序需要预先准备大量用例
- 一些代码分支很难走到
- 靠运气



现代 Fuzz 的特点

- 在有源代码的前提下给代码插桩
- 基于代码覆盖率反馈的 Fuzz
- 把那些能产生新的覆盖路径的用例给保存下来
- 与各种 Sanitizer 相结合（如ASAN、UBSAN、MSAN、TSAN 等等）
- 知名的工具有 AFL、LibFuzzer、HongFuzz 等



FFmpeg 介绍

PART ONE

介绍

FFmpeg是一套目前非常流行的的开源计算机程序。

它提供了录制、播放、转换以及流化音视频的完整解决方案。

目前有非常多的视音频软件或是视频网站、手机 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。

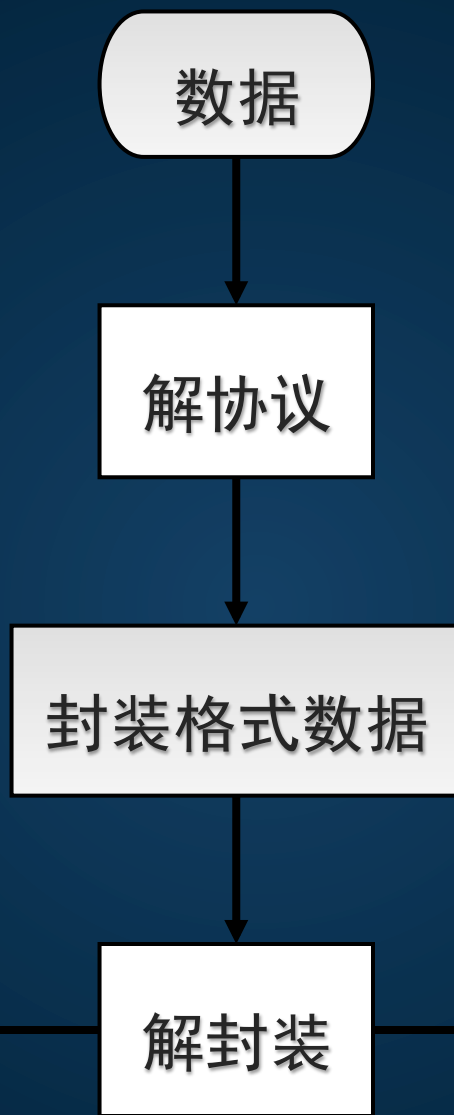


视音频技术简介

视音频技术主要包含以下几点：封装技术，视频压缩编码技术以及音频压缩编码技术。如果考虑到网络传输的话，还包括流媒体协议技术。



视音频流解码流程



音频压缩数据

音频解码

音频原始数据

视频压缩数据

视频解码

视频原始数据

视音频同步



已有的研究成果

PART TWO

Google——FFmpeg and a thousand fixes

Mateusz Jurczyk and Gynvael Coldwind

12年开始，使用突变算法，2000个 core，

样本数据来源于在线的样本库 samples.mplayerhq.hu

以及FFmpeg 自己的 FATE test suite

```
→ SRC git:(master) x git log | grep Jurczyk | grep -c Coldwind  
1524  
→ SRC git:(master) x █
```

Emil Lerner和Pavel Cheremushkin

- 选择了 fuzz FFmpeg 的协议
- American Fuzzy Lop
- Overwrite network operation in FFMpeg
- 阅读协议的源代码

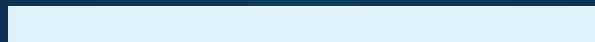
发现的漏洞

Http 协议客户端 RCE、 RTMP 协议 RCE、本地文件包含漏洞



Fuzz 准备

PART THREE



一、寻找合适的攻击面

Codec

google 已经 Fuzz 过了，OSS-Fuzz 上也有 FFmpeg 的 project，里面的 fuzz target 就是针对的 codec

Muxer and Demuxer

Google 也有 Fuzz（通过查看 git log）

Protocol

Emil Lerner和Pavel Cheremushkin使用 AFL Fuzz，改用 libFuzzer

LibFuzzer

- Google 工程师2015年开发，基于 LLVM 的 clang 实现
- 和需要被 Fuzz 的库链接在一起，通过一个特殊的 Fuzz 入口点将

Fuzz 输入喂给 library

- 基于代码覆盖率的反馈来生成新的用例和丢弃无用的用例
- In-process Fuzz, 速度最高能达到 AFL 的几十倍



二、如何 Fuzz 网络协议？

libFuzzer 只能从 buffer 喂输入

重载所有的 socket 相关的操作

已有的轮子——preeny (written by Shellphish)

`LD_PRELOAD=/path/to/desock.so`

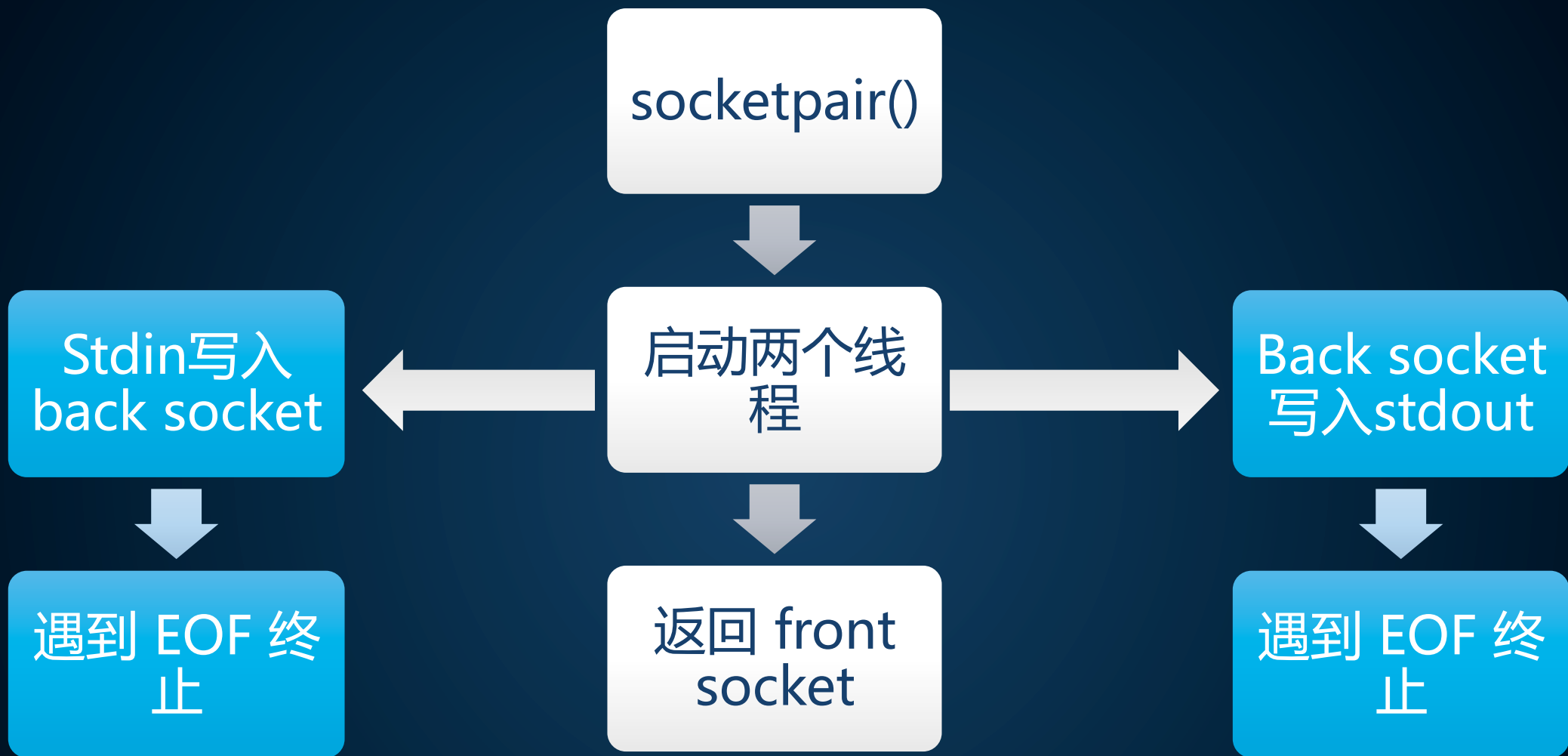


Preeny中 desock.c实现原理

重载了socket , accept , accept4 , bind , listen , connect
等函数。

通过 LD_PRELOAD 环境变量预先加载desock.so文件，覆
盖libc 中的同名函数





定制 preeny

libFuzzer 在fuzz target 启动之初就把所有的数据喂给 stdin ,

在 desock.c 里面 socket 函数是启动一个线程 , 把所有数据从 stdin 写入了

back socket , 等这个线程结束了再往下运行。

针对 ftp 这种需要开启两个 socket 的连接的 , 原本的 preeny 代码不适用。如果

检测是第二次socket 连接 (ftp data port)就往 stdin 里面喂一堆随机数据。



三、Fuzz 目标

针对 rtmp 协议进行 Fuzz

编写一个客户端程序，接受从 rtmp 服务器传来的视频流

测试 avformat_open_input() 这个函数



三、收集样本

- 编写一个 FFmpeg 的程序，从 rtmp 服务器获取视频流，使用 wireshark 抓包
- 寻找不同的 rtmp 源，调整客户端程序的参数来获取不同的 corpus 样本
- 自己写一个 rtmp 协议的字典。



碰到的问题

PART FOUR

一、编译链接

- 尽量使用静态编译
- 利用 pkg-config进行静态链接
- 链接的时候同一个目录下不要既有动态链接库又有静态链接库
(这样即使 -l:libavformat.a 强制指定了链接静态库，运行的时候仍然会报 .so文件未找到)



二、数据包长度过短导致阻塞

- 服务端和客户端有一个握手的过程
- Server 向 Client 发送的三个握手包总计3073个字节
- Server RTMP Version 大于3，client会对握手数据计算 digest 进行校验
- 修改源代码，注释掉校验相关的代码

```
if ((size < 3073) {  
    fprintf(stderr, "data length too small. Exit.\n");  
    return 0;  
}
```


三、数据包过大导致阻塞

- 问题出在 preeny 的 desock.c 里面
- 一股脑将 libFuzzer 提供的 data 从 stdin 写入到 back socket, 导致 socket 的 write buffer 写满, 阻塞在 write 函数里面
- 设定 libFuzzer 的 max_len 参数为 40000

```
n = 0;  
while (n != total_n) {  
    r = write(to, read_buf, total_n - n); // 这里可能阻塞
```

四、彻底解决网络协议的阻塞问题

- 设置 timeout 参数？比较难以控制阈值，导致 Fuzz 速度大大降低
- 设置 interrupt_callback 回调函数
- 官方没有提供接口将网络协议的读写操作设置成非阻塞的。手动修改源代码，

添加 AV_IO_NONBLOCK

```
#ifdef FUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION
if ((ret = ffurl_open_whitelist(..., AVIO_FLAG_READ_WRITE |
AVIO_FLAG_NONBLOCK, ...)) < 0)
#else
if ((ret = ffurl_open_whitelist(..., AVIO_FLAG_READ_WRITE,...) < 0)
#endif
```

FUZZ 成果

PART FIVE

libFuzzer 运行情况

- 初始 cov 就达到了一个较高的水准
- 一分钟就跑出了 crash
- 大量的heap buffer overflow越界读
- 极少是Segmentation Fault
- ffmpeg 的 rtmp 协议中有两个函数处理 packet 的时候没有验证 length 是

否越界



CVE-ID

CVE-2017-11665

[Learn more at National Vulnerability Database \(NVD\)](#)

• Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings

Description

The ff_amf_get_field_value function in libavformat/rtmppkt.c in Ffmpeg 3.3.2 allows remote RTMP servers to cause a denial of service (Segmentation Violation and application crash) via a crafted stream.


References

Note: [References](#) are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- [MISC:https://gist.github.com/7d94dda50856e707e1c92d068bbc244e](https://gist.github.com/7d94dda50856e707e1c92d068bbc244e)
- [BID:100017](#)
- [URL:http://www.securityfocus.com/bid/100017](http://www.securityfocus.com/bid/100017)

FFmpeg 'libavformat/rtmppkt.c' Denial of Service Vulnerability

Bugtraq ID: 100017
Class: Boundary Condition Error
CVE: CVE-2017-11665
Remote: Yes
Local: No
Published: Jul 27 2017 12:00AM
Updated: Jul 27 2017 12:00AM
Credit: JunDong Xie from Ant-financial Light-Year Security Lab.



官方修复方式

- avformat/rtmppkt: Convert ff_amf_tag_size() to bytestream2
- avformat/rtmppkt: Convert ff_amf_get_field_value() to bytestream2
- Bytestream2 : 对读写数据的一种封装，会检查读写是否越界



Code

Pull requests 1

Projects 0

Insights

avformat/rtmppkt: Convert ff_amf_get_field_value() to bytestream2

Fixes: out

FFmpeg / FFmpeg

Watch

803

Found-by:

Signed-off-by:

Code

Pull requests 1

Projects 0

Insights

avformat/rtmppkt: Convert ff_amf_tag_size() to bytestream2

Fixes: out of array accesses

Fixes: crash-9238fa9e8d4fde3beda1f279626f53812cb001cb-SEGV

Found-by: JunDong Xie of Ant-financial Light-Year Security Lab

Signed-off-by: Michael Niedermayer <michael@niedermayer.cc>

参考资料

- CVE: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-11665>
- LibFuzzer: <https://llvm.org/docs/LibFuzzer.html>
- FFmpeg 雷霄骅的博客: <http://blog.csdn.net/leixiaohua1020/article/details/15811977/>
- Preeny: <https://github.com/zardus/preeny>
- google 的 OSS-Fuzz: <https://github.com/google/oss-fuzz/>



THANKS

