

EVOLUTION OF ANDROID EXPLOITS FROM A STATIC ANALYSIS TOOLS PERSPECTIVE



VB2014: Anna Szalay & Jagadeesh
Chandraiah

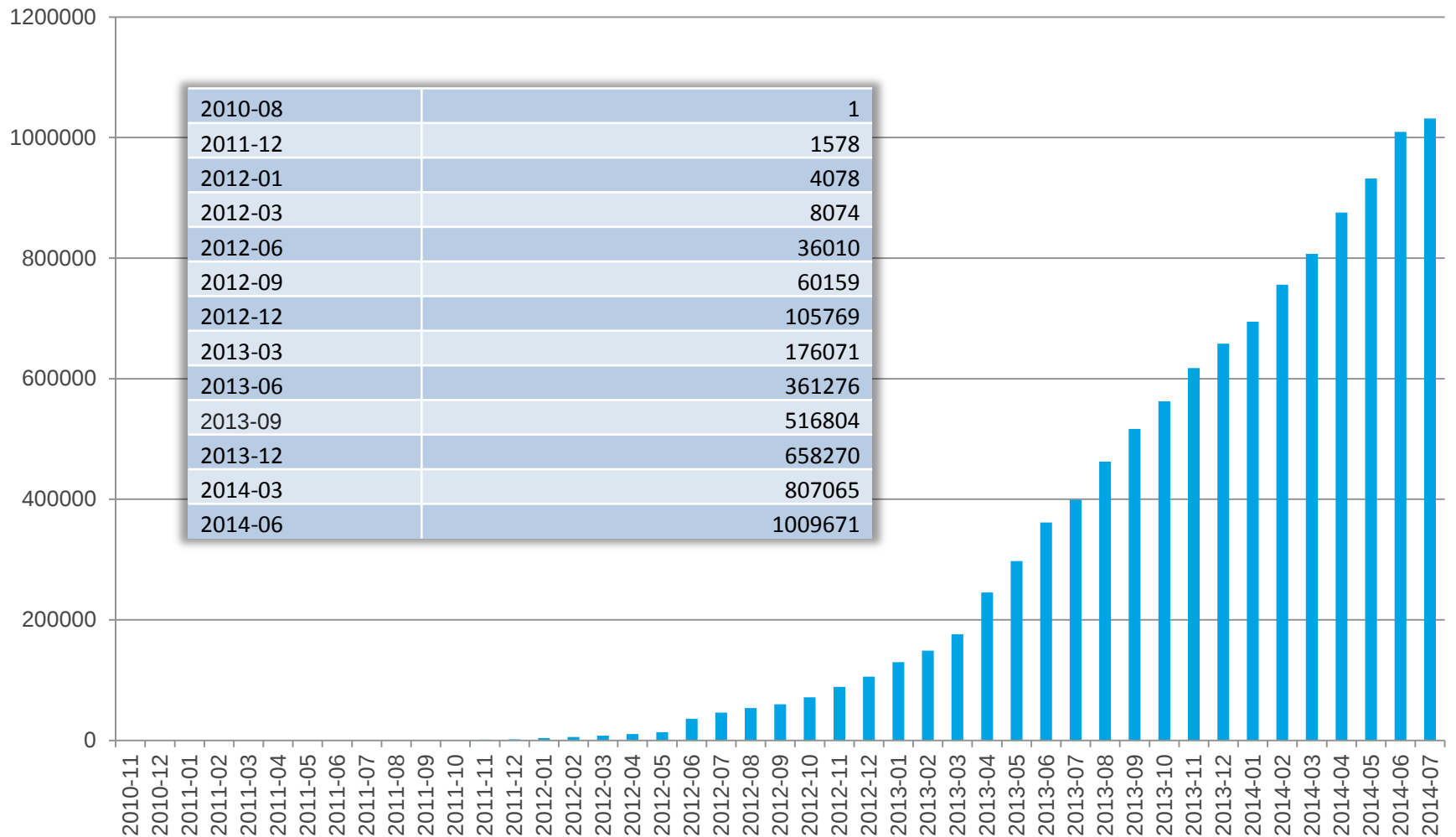
Email {anna.szalay,
jagadeesh.chandraiah}@
sophos.com

Devices cumulative

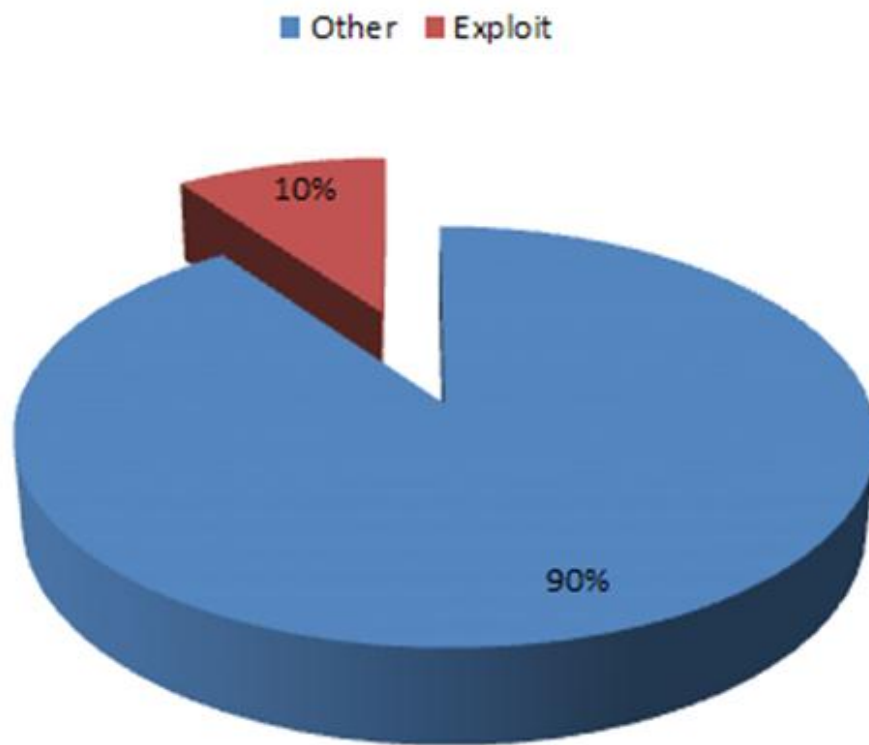
Operating System	2013	2014	2015
Android	879,821	1,170,952	1,358,265
Windows	325,127	339,068	379,299
iOS/Mac OS	241,416	286,436	324,470
Others	873,194	683,519	565,186
Total	2,319,559	2,479,976	2,627,221

Worldwide Device Shipment by Operating System (Thousands of Units).
Gartner (March 2014). <http://techcrunch.com/2014/03/27/gartner-devices-forecast-2014>.

Malware Samples cumulative



Malware exploits samples share



Exploits overview

Android Vulnerabilities Exploitation Timeline

Q4 2010

Android WebKit browser exploit

Q4 2010

Android Data Stealing Vulnerability

Q1 2011

Android Local Root Exploit aka “Rage against the cage” or Lotoor exploit

Q3 2011

Android ClientLogin Protocol Vulnerability

Q3 2011

Android Gingerbreak root exploit

2012

...

Q2 2013

DEX2JAR exploitation

Q2 2013

“Master Key” vulnerability

Q2 2013

“Extra Field” vulnerability

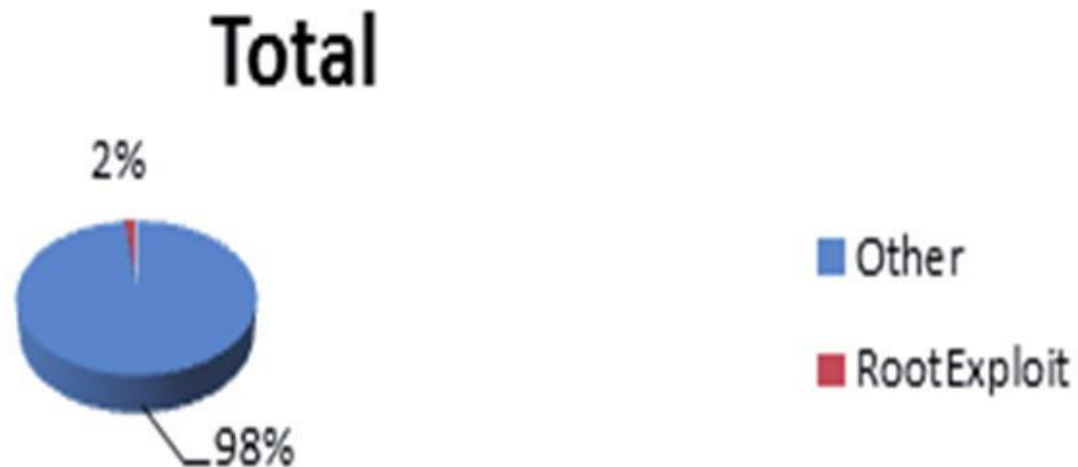
Exploits overview

Q1 2011

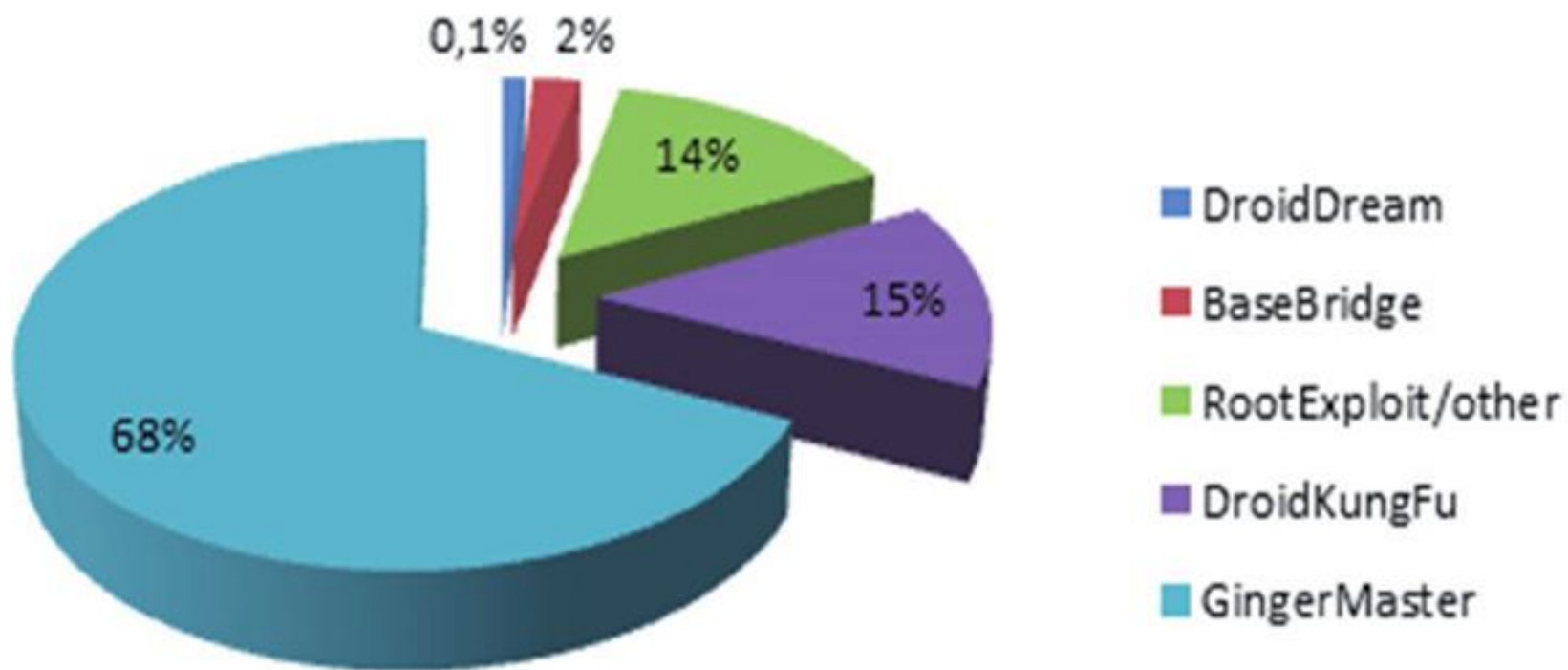
Android Local Root Exploit aka “Rage against the cage” or Lotoor exploit

- ☐ **Andr/DroidRt aka DroidRoot (ELF EXECUTABLE)**
- ☐ **Andr/DroidD aka DroidDream**
- ☐ **Andr/KongFu aka DroidKungFu**

Malware root exploit samples share



Share of root-exploit-based malware samples by threat



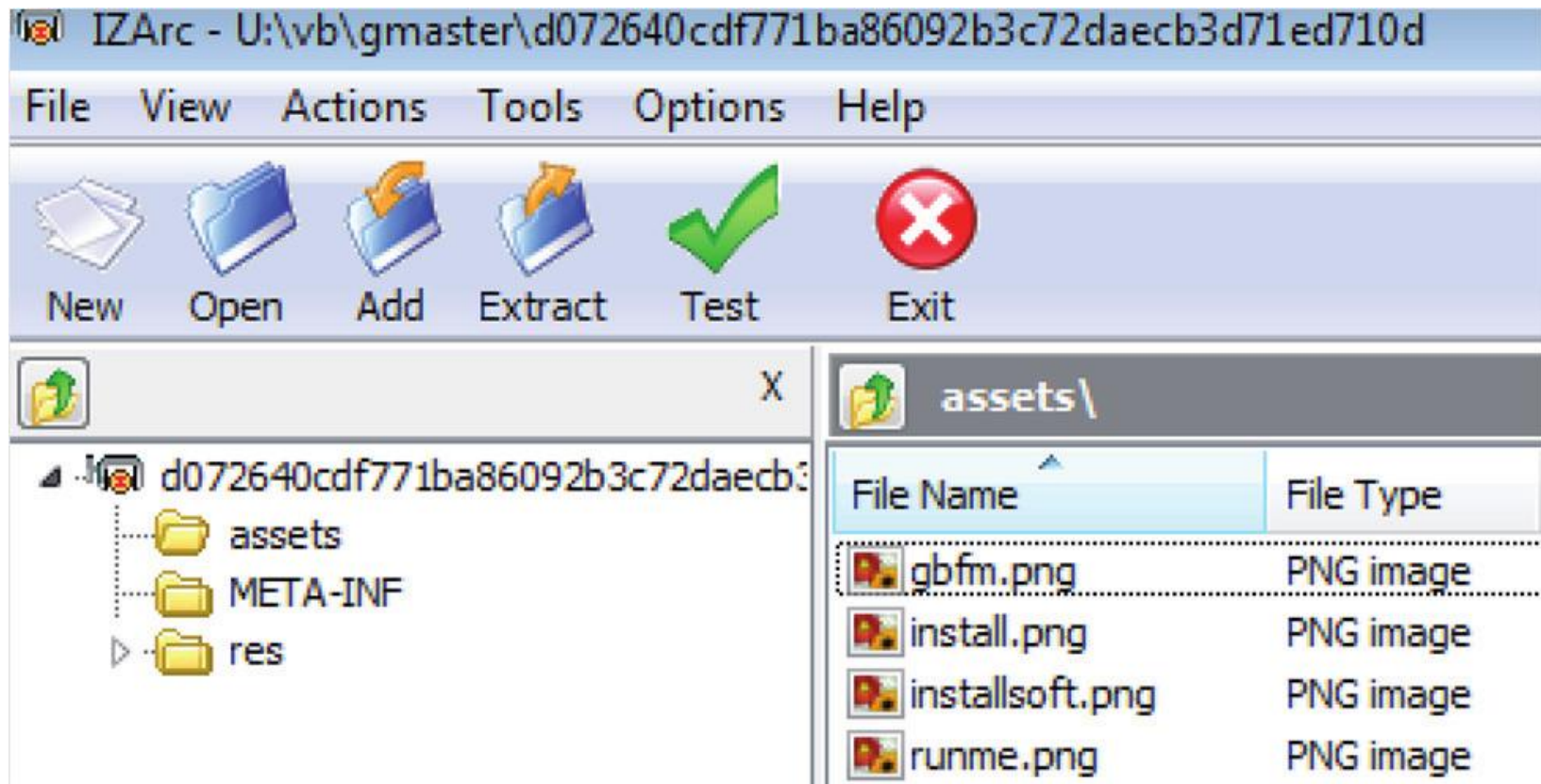
Exploits overview

Q3 2011

Android Gingerbreak root exploit

- ❑ **CVE-2011-1823**
- ❑ **allows local users to execute arbitrary code and gain root privileges via a negative index that bypasses a maximum-only signed integer check on the DirectVolume::handlePartitionAdded method, which triggers memory corruption**
- ❑ **Andr/Gmaster-A aka GingerMaster**

Android Gingerbreak root exploit : Inside GMaster APK, exploit code in a picture file.



Android Gingerbreak root exploit : Inside Gmaster APK, exploit code in a picture file, IDA image

IDA - U:\vb\gmaster\assets\gbfm.idb (gbfm.png)

File Edit Jump Search View Debugger Options Windows Help

Library function Data Regular function Unexplored Instruction External symbol

IDA View-A Names window Hex View-A Structures Enums Imports Exports

```
00 28      CMP     R0, #0
01 D0      BEQ     loc_9B86
FF F7 CD FF  BL     sub_9B20
;
loc_9B86  ; CODE XREF: sub_9B4C+1E↑j
          ; sub_9B4C+26↑j ...
C0 48      LDR     R0, =(aGingerbreakHoneybombAndroid2_233_0Softb - 0x9B90)
C0 4D      LDR     R5, =(__sf_ptr - 0xB108)
C1 4C      LDR     R4, =(dword_B020 - 0x9B98)
78 44      ADD     R0, PC ; "\n[**] Gingerbreak/Honeybomb -- android"...
FF F7 70 E8  BLX     puts
C0 48      LDR     R0, =(a(c)20102011TheAndroidExploitCrew_AllRig - 0x9B9A)
7C 44      ADD     R4, PC ; dword_B020
78 44      ADD     R0, PC ; "[**] (C) 2010-2011 The Android Exploit "...
FF F7 6A E8  BLX     puts

00001B96 00009B96: sub 9B4C+4A
```

Output window

Autoanalysis subsystem has been initialized.

IDC

AU: idle Down Disk: 69GB

Exploits overview

Q2 2013

DEX2JAR exploitation : Andr/Obad-A

- ❑ disrupts the conversion of Dalvik bytecode into Java bytecode
- ❑ exploits AndroidManifest.xml processing
- ❑ gains extended Device Admin privileges

Andr/Obad-A : classes.dex obfuscation

complex code encryption

- ☐ all external methods are called via reflection
- ☐ all strings are encrypted, including the names of classes and methods
- ☐ each class has a local descriptor method which obtains the string required for encryption from the locally updated byte array
- ☐ all strings are hidden in this array
- ☐ additional stage of decryption for C&C addresses

Obad obfuscation, example of decompiled by dex2jar dex code output

Java Decompiler - IExtendedNetworkService.class

File Edit Navigate Search Help

02341c8052d5069f4e11d08ee0626e428a20f194_dex2jar.jar

internal.telephony

- IExtendedNetworkService
 - IExtendedNetworkService
 - oCI1C11
 - clearMmiString() : void
 - getMmiRunningText() : CharSequence
 - getUserMessage(CharSequence) : CharSequence
 - setMmiString(String) : void
- system.admin
 - CCOIoll
 - CCIIOcc
 - CICoICCo
 - CICoICo
 - CIIOCClc
 - COOIOII
 - COcCcl
 - CcOCcIcO
 - CoOOoOo
 - CoccOIo

IExtendedNetworkService.class

```
package com.android.internal.telephony;

import android.os.Binder;

public abstract interface IExtendedNetworkService {
    public abstract void clearMmiString();

    public abstract CharSequence getMmiRunningText();

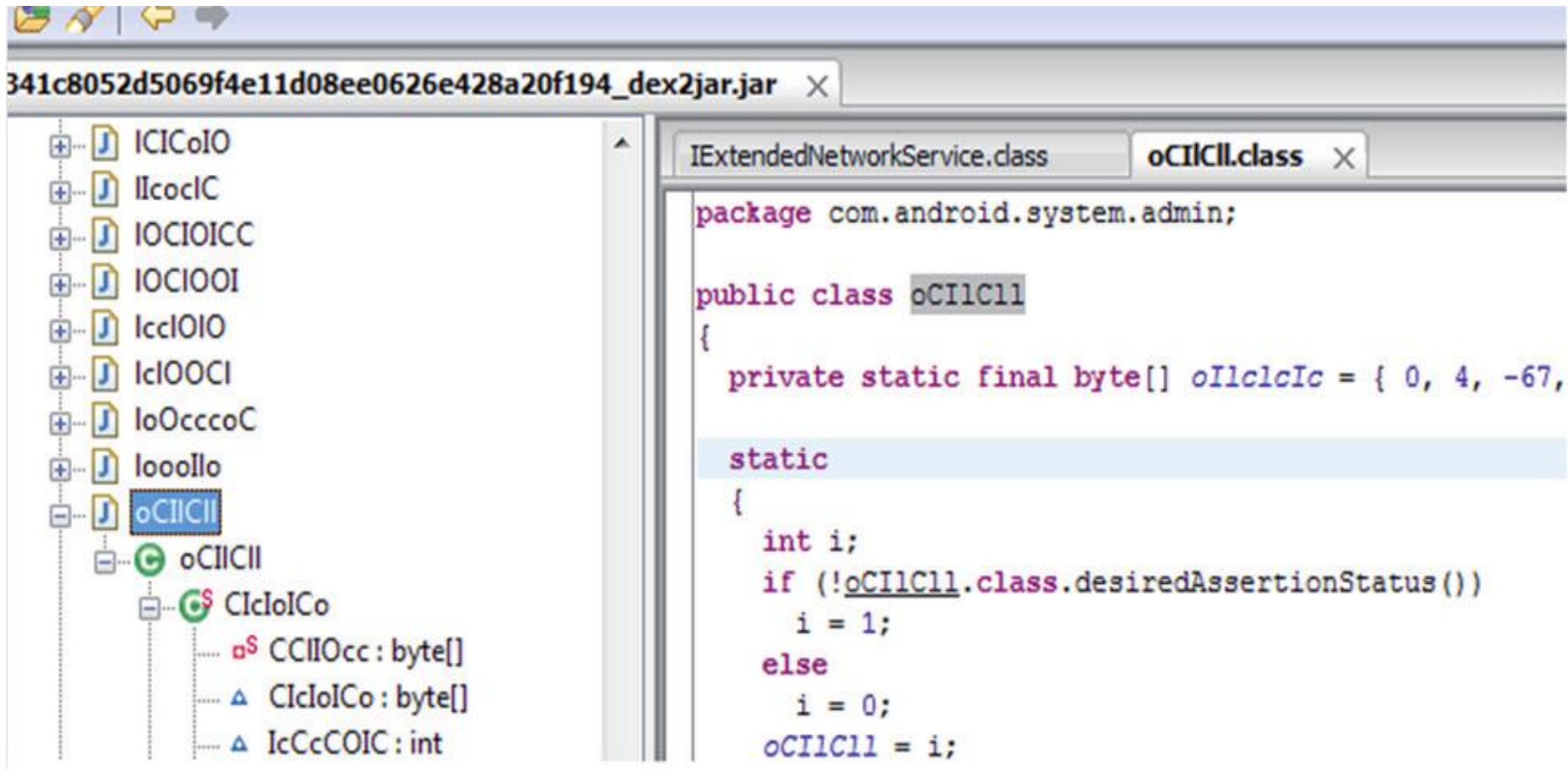
    public abstract CharSequence getUserMessage(CharSequence paramCharSequence);

    public abstract void setMmiString(String paramString);

    public static abstract class oCI1C11 extends Binder implements IExtendedNetworkService {
        private static final byte[] oCI1C11 = { 1, -24, ... };

        public oCI1C11() {
            super(1);
        }
    }
}
```

Obad obfuscation, example of decompiled by dex2jar dex code output



Obad obfuscation, example of decompiled by dex2jar dex code output

Java Decompiler - IExtendedNetworkService.class

File Edit Navigate Search Help

02341c8052d5069f4e11d08ee0626e428a20f194_dex2jar.jar

com.android
internal.telephony
IExtendedNetworkService
IExtendedNetworkService
oCIIClI
clearMmiString(): void
getMmiRunningText(): CharSe
getUserMessage(CharSequenc
setMmiString(String): void

system.admin
CCOIoll
CCIOcc
CICoICCo
CIdIoICo
CIIOCcIc
COOIOL
COcCcd
CcOCOLcO
CoOOoOo
CoccOlo
CoooOIO
ICICcOCc
ICOCOLc
ICOLoCI
ICcIllo
ICcICcOL
ICOLoCI

IExtendedNetworkService.class

```
package com.android.internal.telephony;

import android.os.Binder;

public abstract interface IExtendedNetworkService extends IInterfa
{
    public abstract void clearMmiString();

    public abstract CharSequence getMmiRunningText();

    public abstract CharSequence getUserMessage(CharSequence paramCh

    public abstract void setMmiString(String paramString);

    public static abstract class oCIIClI extends Binder
    implements IExtendedNetworkService
    {
        private static final byte[] oCIIClI = { 1, -24, 32, -15, 15, 7

        public oCIIClI()
        {
            attachInterface(this, oCIIClI(0, 43, -9));
        }

        private static String oCIIClI(int paramInt1, int paramInt2, in
        {
            int i = paramInt3 + 108;
            byte[] arrayOfByte1 = oCIIClI;
        }
    }
}
```

Exploits overview

Q2 2013

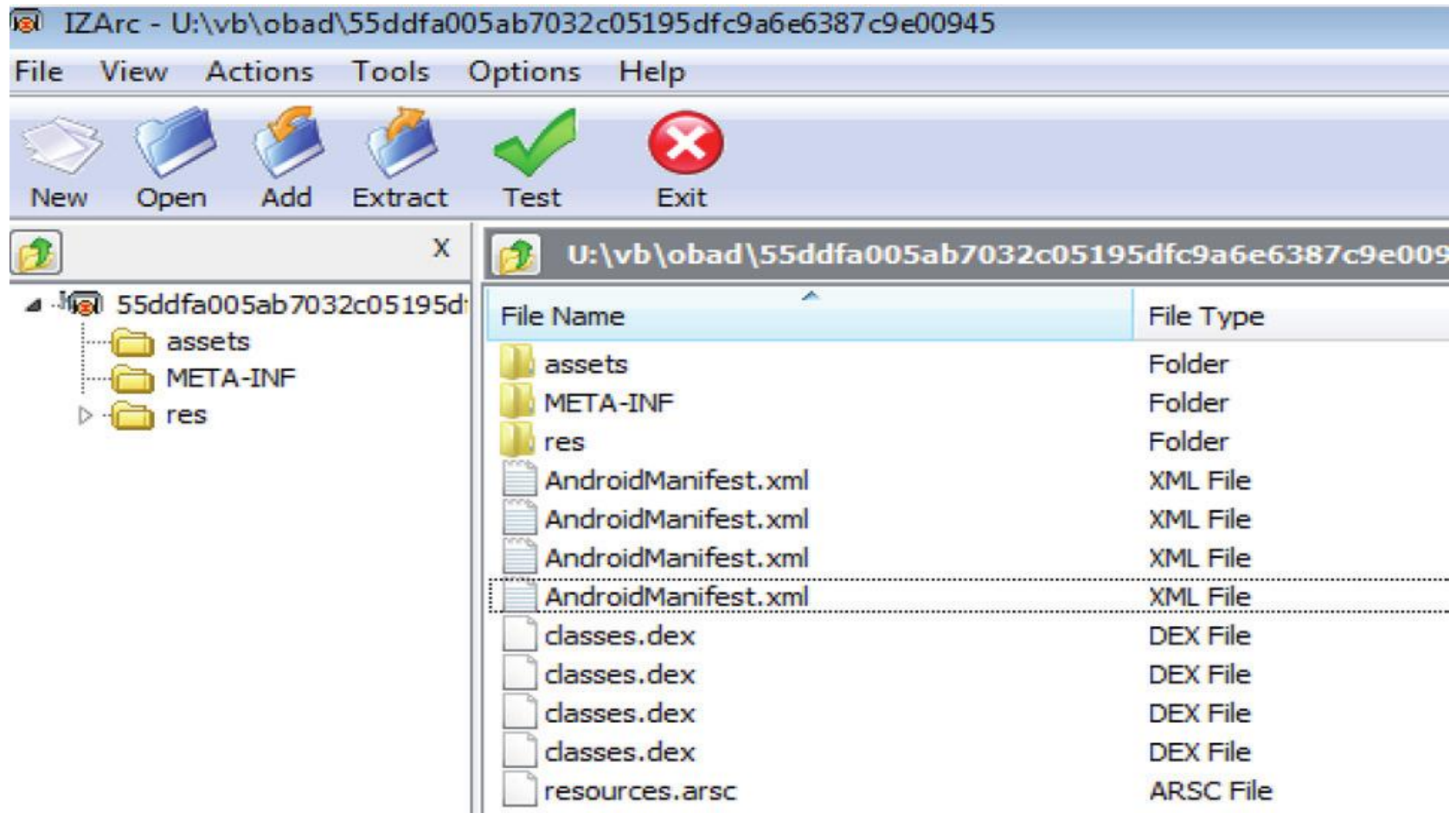
‘Master Key’ vulnerability

- ❑ **CVE-2013-4787**
- ❑ **reported to affect 99% of devices**
- ❑ **apk signature ‘compromised’**
- ❑ **allows to replace installed app with the compromised ‘update’**

“Master Key” vulnerability : Andr/MstrKey-A

- ☐ exploited apk file has more than one classes.dex and AndroidManifest.xml files
- ☐ dodgy manifest file shows permissions that won't be seen when we look at its actual fingerprint

Inside 'Master Key' malware, example of multiple files.



Exploits overview

Q3 2013

‘Extra Field’ vulnerability

- ☐ allows to bypass code verification
- ☐ uses an extra field during an APK verification as archive
- ☐ based on signed-unsigned integer mismatch
- ☐ exploits fundamentals of the Android APK as a ZIP archive with the some special object fields

Example of malware exploiting the 'Extra Field' vulnerability, showing a changed field

Hex Workshop - [U:\vb\AndrExtraFld\Q-033196\samples\75416e8dbf256242ccd0e3353430915d7cffb74b]

File Edit Disk Options Tools Window Help

ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	
00000294	00	00	00	00	00	00	00	00	00	0B	00	FD	
000002A0	FF	63	6C	61	73	73	65	73	2E	64	65	78	.classes.dex
000002AC	0A	30	33	35	00	51	8C	10	4D	02	BD	72	.035.Q..M..r
000002B8	18	D5	0A	8C	F4	4F	FC	3E	6F	7B	DA	52	0.>o{.R

75416e8dbf...

Structures zip structures (zip-format.hsl)

Member	Value (dec)	Value (hex)
00000283 struct LocalFileHe...	{...}	
00000283 char Signature[4]	PK	
00000287 WORD VersionN...	20	1400
00000289 WORD GeneralP...	8	0800
0000028B COMPRESSION_...	STORED (0)	0000
0000028D DOSTIME LastM...	12:21:40	B462
0000028F DOSDATE LastM...	06/09/2012	2641
00000291 DWORD Crc32	0	00000000
00000295 DWORD Compre...	0	00000000
00000299 DWORD Uncom...	0	00000000
0000029D WORD FileNam...	11	0B00
0000029F WORD ExtraField...	65533	FDFF
000002A1 char FileName[F...	classes.dex	

Compare Re

Type

Milestones in exploiting

Root Exploits

Andr/DroidRt (DroidRoot)
Andr/DroidD (DroidDream)
Andr/Kongfu (DroidKungFu)
Andr/Gmaster (GingerMaster)

DEX2JAR

Andr/Obad (Obad)

APK Signature

Andr/MstrKey (MasterKey)



Evolution of Android Exploits from a Static Analysis Tools Perspective - A Szalay, J Chandraiah

What's Next ?

- **Static Analysis Tools and Techniques**
 - Discuss various Android Static Tools and Techniques
- **Evaluation of Tools against Exploit Samples**
 - Android Master Key Vulnerability
 - Dex Header
 - Unfamiliar Opcodes
 - Decompilation Issues

Why ?

- To verify if tools really meet the requirements
- Highlight issues encountered by analysts
- Identify causes that break various tools
- Suggest best approach for improvement based on the outcome of the research

APK and key elements



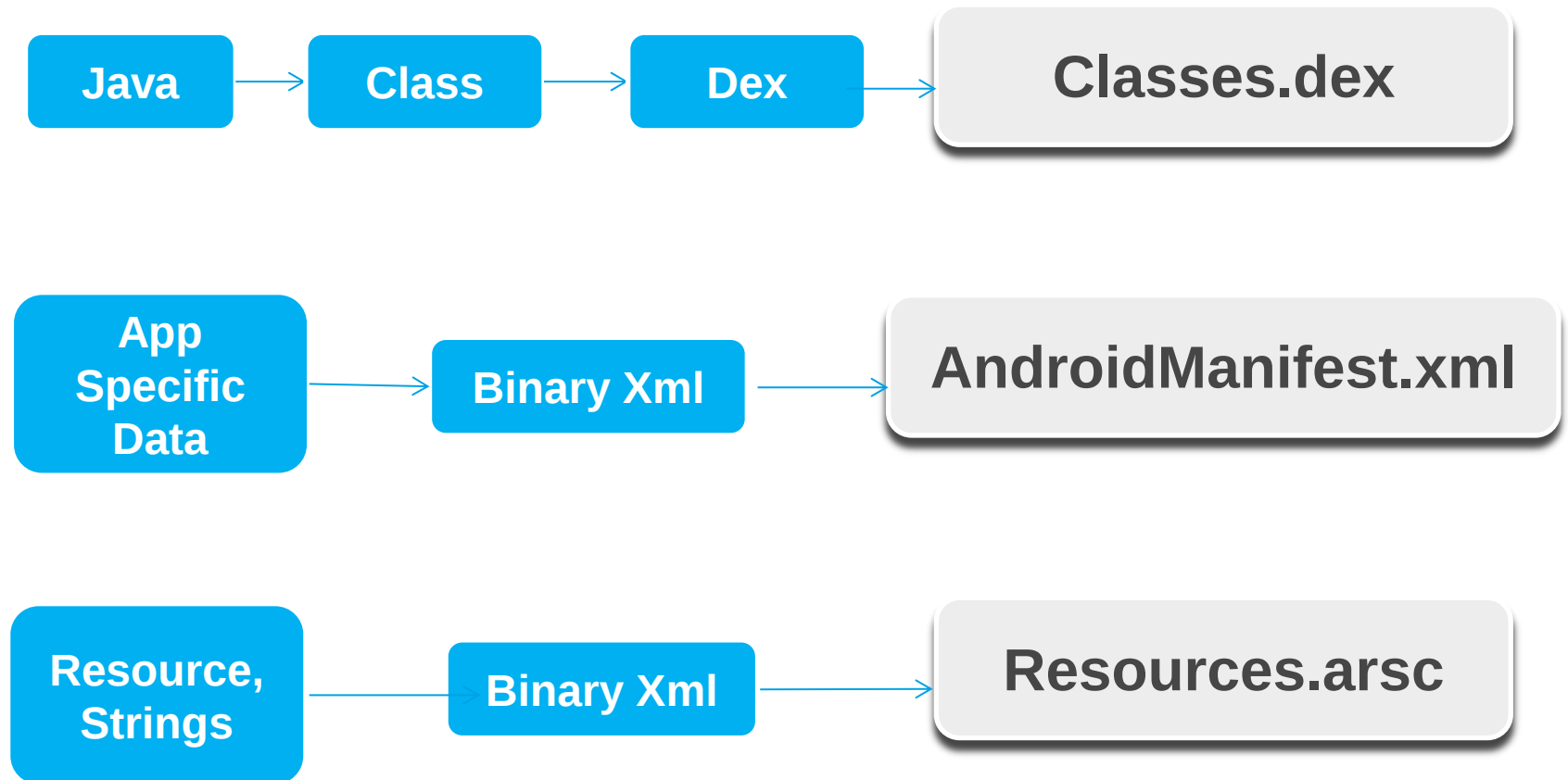
APK

Classes.dex

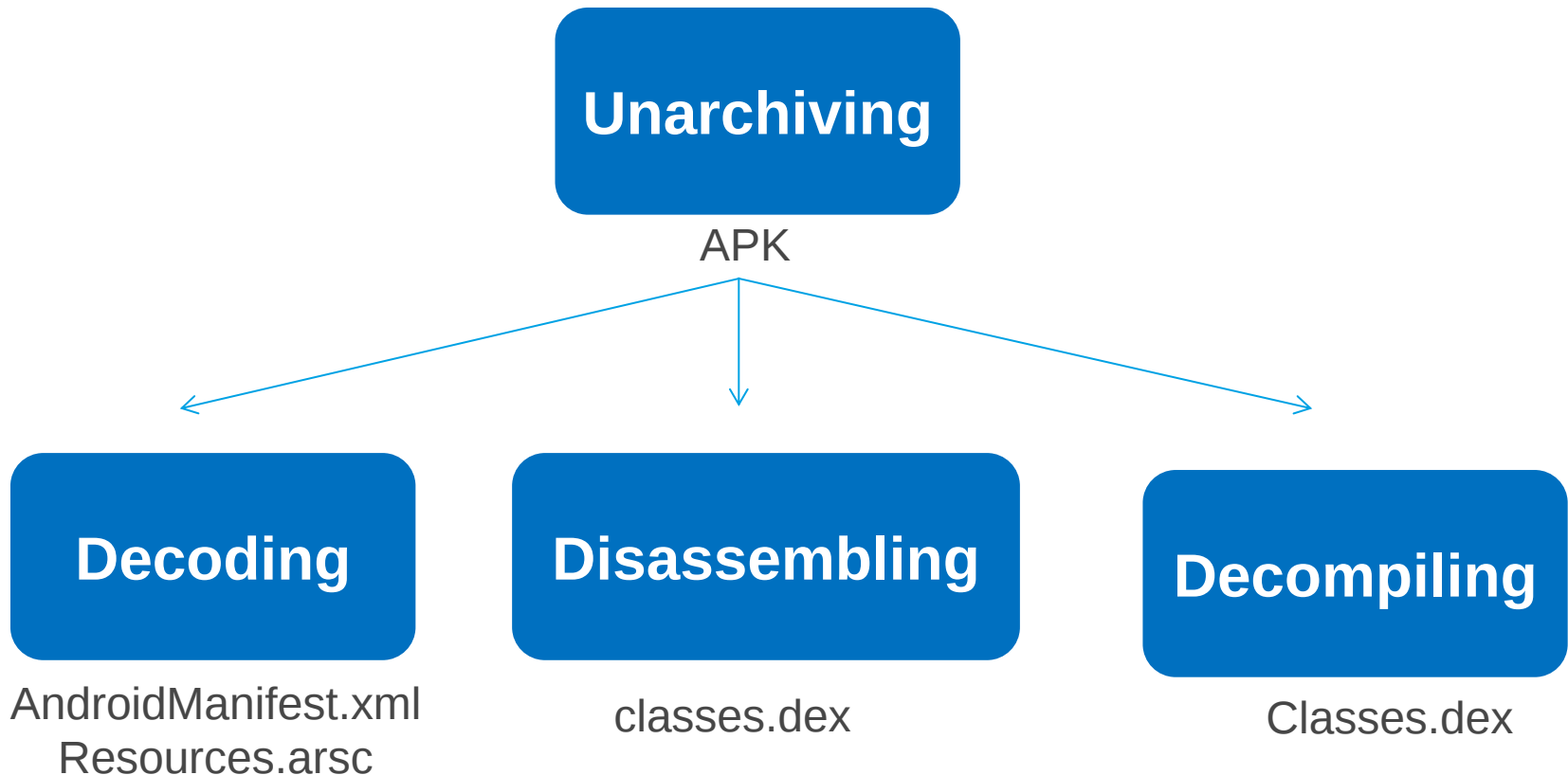
AndroidManifest.xml

Resources.arsc

APK and key elements



Static Analysis Techniques



Static Analysis Tools

- **Unarchiving** – Unzip or any similar archive extraction tools.
- **Decoding** – Apktool, Androguard
- **Decompilation** - Dex2jar, Jdgui, JEB, ded and many other java decompilers.
- **Disassembly** – smali/baksmali, dexdump and IDA pro

Methodology

- Most commonly used analysis technique and Tools used for testing.
- Selected group of popular and relevant Android Exploit samples were used for testing.
 - Andr/MstrKey, Andr/DroidD (DroidDream), Andr/DroidRt, Andr/Obad, Andr/Kongfu (DroidKungfu), Andr/Gmaster (GinMaster)
- Discuss cases that makes analysis difficult
- Also applies to similar non exploit Android samples.

Andr/MstrKey-A- Overwriting?

- Trivial but Important in the case of this Exploit analysis
- Might end up analysing the wrong file

```
inflating: assets/fashion.jpg  
inflating: res/layout/main.xml  
inflating: AndroidManifest.xml  
replace AndroidManifest.xml? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
inflating: res/drawable-hdpi/icon.png  
inflating: res/drawable-ldpi/icon.png  
inflating: res/drawable-mdpi/icon.png  
inflating: classes.dex  
replace classes.dex? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

Andr/MstrKey-A- Duplicate files!

- Watch out if you want to use Apktool d <apk>

2320255	21/07/2013	05:31	assets/fashion.jpg
640	21/07/2013	05:31	res/layout/main.xml
6780	21/07/2013	05:31	AndroidManifest.xml
1404	21/07/2013	05:31	AndroidManifest.xml
1476	21/07/2013	05:31	resources.arsc
7542	21/07/2013	05:31	res/drawable-hdpi/icon.png
2760	21/07/2013	05:31	res/drawable-ldpi/icon.png
4278	21/07/2013	05:31	res/drawable-mdpi/icon.png
90080	21/07/2013	05:31	classes.dex
4908	21/07/2013	05:31	classes.dex
635	21/07/2013	05:31	META-INF/MANIFEST.MF
688	21/07/2013	05:31	META-INF/CERT.SF
1007	21/07/2013	05:31	META-INF/CERT.RSA

Andr/DroidD - Dex header

- IDA Pro 6.4adv default installation displayed corrupt message



Andr/DroidD - Dex header

Offset	Description
0x0 – 0x7	DEX_FILE_MAGIC

DEX_FILE_MAGIC = "dex\n<Version number>\0"

- 0x64 0x65 0x78 0x0a 0x30 0x33 0x36 0x00 = “**dex.036**”
- Version ‘036’ – Current version (4.x)
- Version ‘035’ – older api level 13 and earlier , most malware had this version number in our DB.

<http://www.strazzere.com/blog/2013/02/loose-documentation-leads-to-easy-disassembler-breakages>

Unfamiliar Opcodes

- Familiar technique employed in PC world
- Simple but still effective way
- Lot of research already done and now see them being used
- Packers, malware use to break tools and slow analysis

Unfamiliar Opcodes

- Dex2jar and baksmali v 1.4.1* failed to work

Caused by: java.lang.RuntimeException: **opcode format for 64 not found!**

at

com.googlecode.dex2jar.reader.OpcodeFormat.get(OpcodeFormat.java:362)

at

... 8 more

```
unknown opcode encountered - 40. Treating as nop.
```

```
UNEXPECTED TOP-LEVEL EXCEPTION:
```

```
org.jf.dexlib.Util.ExceptionWithContext: Index: 22789, Size: 269
```

```
at org.jf.dexlib.Util.ExceptionWithContext.withContext(ExceptionWithContext.java:54)
```

```
at org.jf.dexlib.IndexedSection.getItemByIndex(IndexedSection.java:77)
```

```
at org.jf.dexlib.Code.InstructionWithReference.lookupReferencedItem(InstructionWithReference.java:79)
```

```
at org.jf.dexlib.Code.InstructionWithReference.<init>(InstructionWithReference.java:57)
```

```
at org.jf.dexlib.Code.Format.Instruction22c.<init>(Instruction22c.java:59)
```

```
at org.jf.dexlib.Code.Format.Instruction22c.<init>(Instruction22c.java:40)
```

```
at org.jf.dexlib.Code.Format.Instruction22c$Factory.makeInstruction(Instruction22c.java:103)
```

*Works in baksmali version 2.03

Unfamiliar Opcodes

- **40** - unused_40*
- **64** - Reads the char static field identified by the field_id into VX.
- **6400 0200** - sget-byte v0, Test3.bs1:B // field@0002
Reads byte field@0002 (entry #2 in the field id table) into v0

*src -http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

Decompilation

- Decompilation is preferred method of analysis.
- Dex2jar followed by jdgui/jad is the most used method.
- Junk insertion, obfuscation , not maintained.

```
}  
  
// ERROR //  
public final void uncaughtException(java.lang.Thread paramThread, java.lang.Throwable paramThrowable)  
{  
    // Byte code:  
    // 0: new 80    java/lang/StringBuilder  
    // 3: dup  
    // 4: bipush 254
```

Caused by: java.lang.ArrayIndexOutOfBoundsException: 105
at
com.googlecode.dex2jar.v3.V3CodeAdapter.visitMoveStmt(V3CodeAdapter.java:554) at

Outcome ?

- Popular tools aren't enough for complete analysis
- Tools update still catching up with research
Implementation
- Support for tools maintenance and development by
Android/Google
- IDE for malware analysis ?????
- Better to combine with dynamic Analysis

...Questions...

