

Android Packer



facing the challenges, building solutions

Rowland YU

Senior Threat Researcher

Virus Bulletin 2014

SOPHOS

What is Android Packer?

Android packers are able to
encrypt an original classes.dex file,
decrypt the dex file to memory at runtime,
and then execute via **DexClassLoader**.

What is Particular about Android Packer?

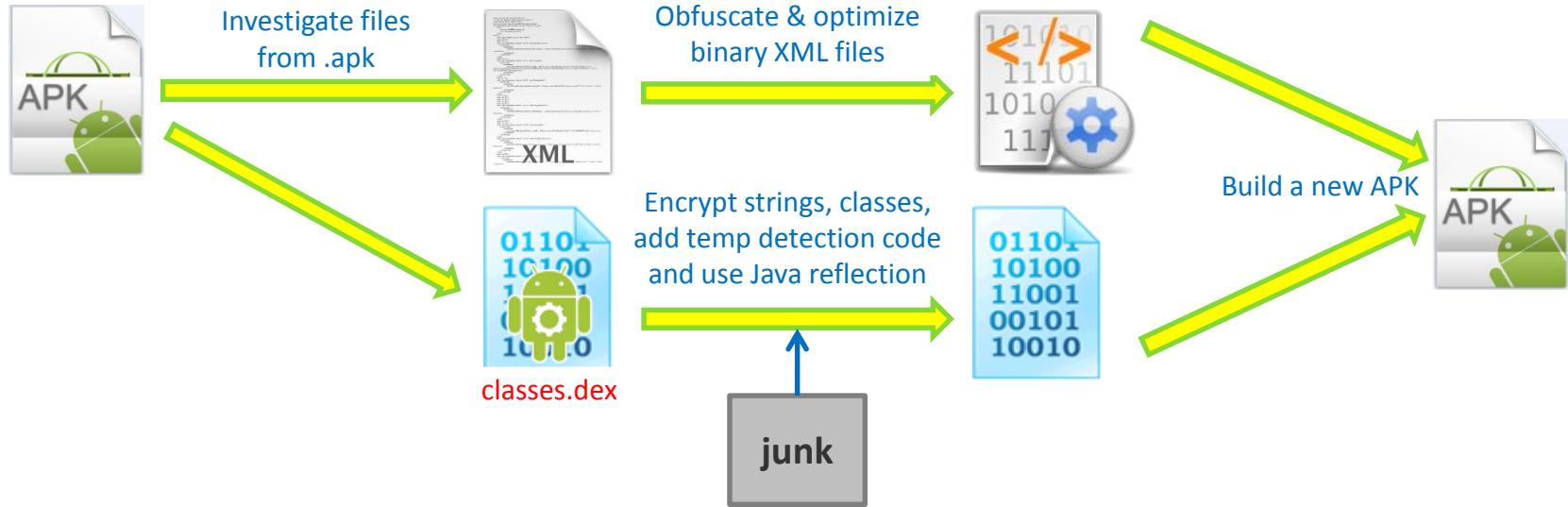
- Difference between Obfuscation and Packer
- Popular packer services
- Growth of packed threats
- Opening the black box of Android Packers

How to Evade Android Packer?

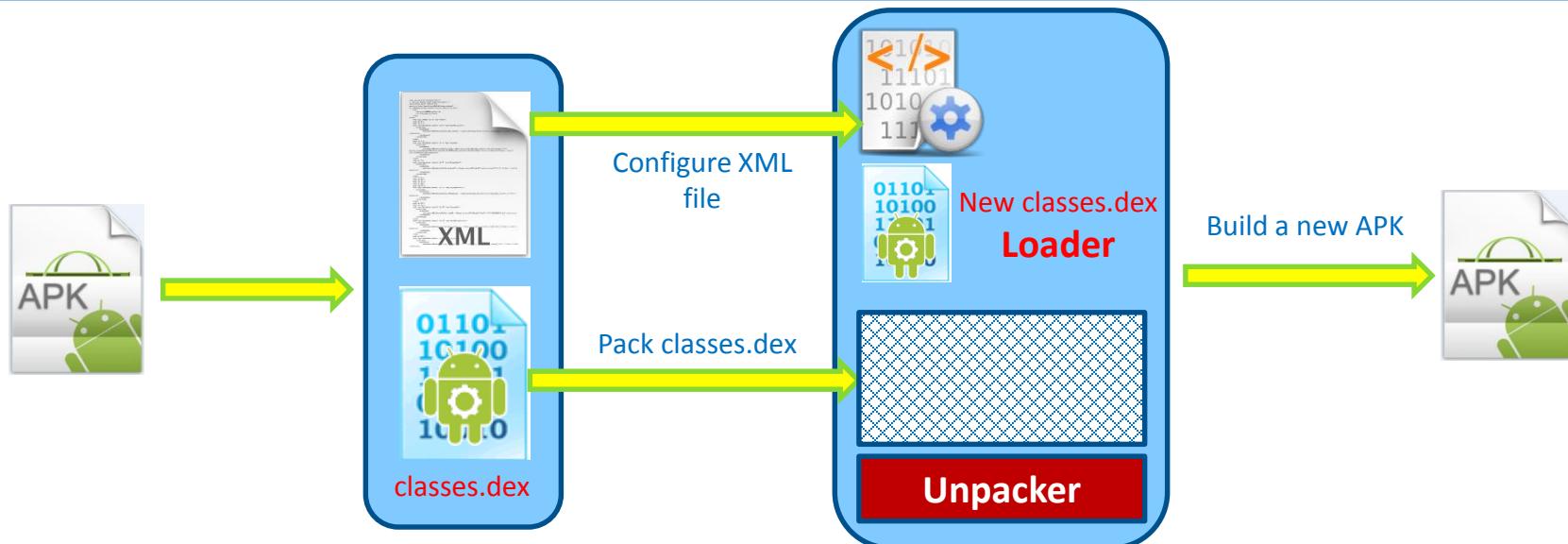
- Challenges
- Solutions
- Demonstration
- Summary
- The Future

Obfuscation

Difference between Obfuscation and Packer



Packer



Popular packer services

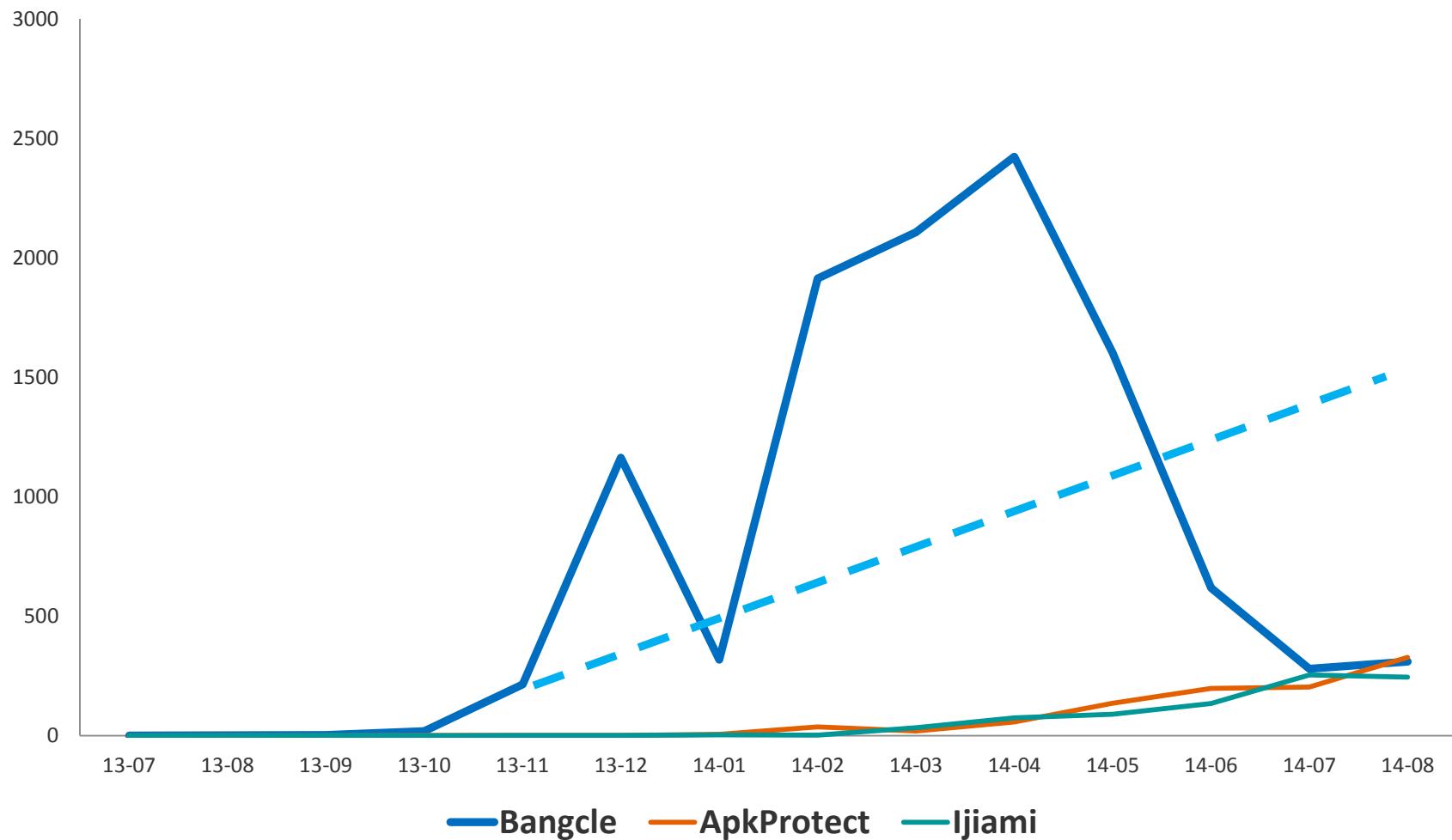


BANGCLE梆梆

爱加密 Ijiami

- Protecting legitimate applications in loss of intellectual property
- Online Android packing services
- Anti-Tamper, Anti-Decompiler, Anti-Runtime injection, and Anti-Debug
- Employing virus scan engines to prevent malware being packed

Growth of Packed Threats



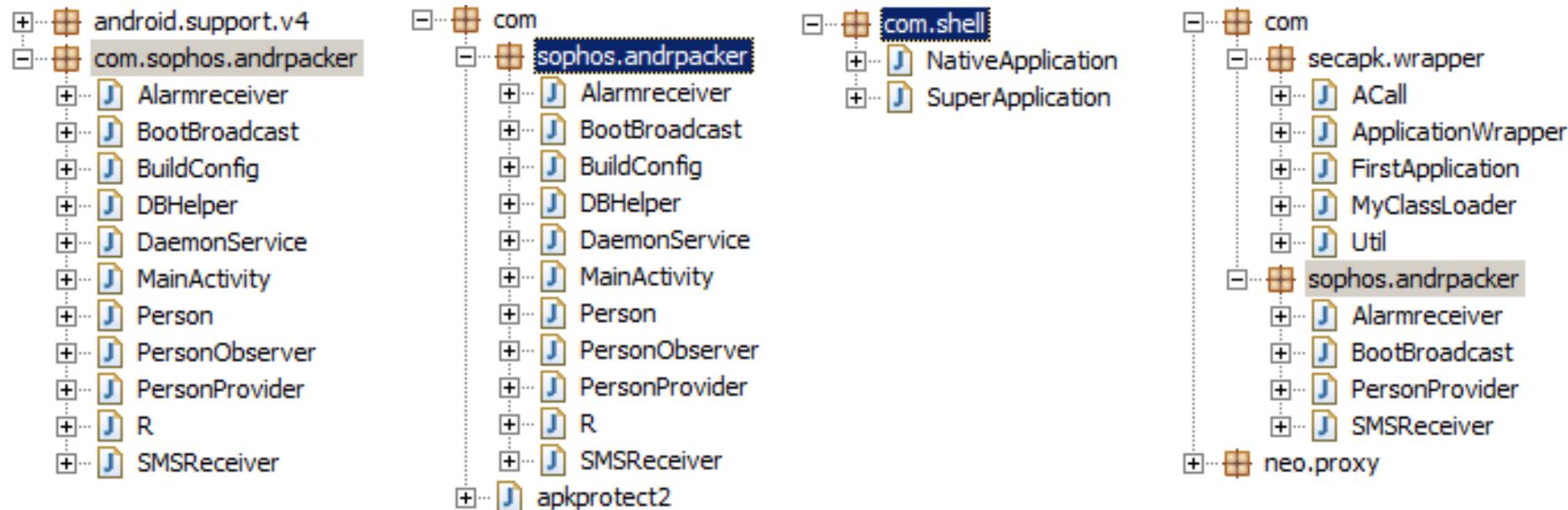
Opening the Black Box of Android Packers

Pack Provider	Added File	Comments
ApkProtect	lib/armeabi/libapkprotect2.so	// ARM shared native library binary
Bangcle	assets/meta-data/manifest.mf	// APK manifest file
	assets/meta-data/rsa.pub	// Signature file
	assets/meta-data/rsa.sig	// The real signature file with certificate
	assets/bangcle_classes.jar	// Encrypted original classes.dex file
	assets/bangcleplugin/collector.dex	// Bangcle information collector plugin
	assets/bangcleplugin/container.dex	// Bangcle Implementation plugin
	assets/bangcleplugin/dgc	// Bangcle plugin logfile
	assets/com.sophos.andrpacker	// ARM executable file
	assets/com.sophos.andrpacker.x86	// x86 executable file
	assets/libsecexe.x86.so	// x86 shared native library binary
	assets/libsecmain.x86.so	// x86 native main binary
	lib/armeabi/libsecexe.so	// ARM shared native library binary
	lib/armeabi/libsecmain.so	// ARM native main binary
Ijiami	META-INF/signed.bin	// Ijiami signed binary file
	META-INF/af.bin	// Ijiami binary file
	META-INF/sdata.bin	// Ijiami RSA signature file
	assets/ijiami.dat	// Encrypted original Apk file
	lib/armeabi/libexecmain.so	// ARM JNI load/unload native binary
	lib/armeabi/libexec.so	// ARM shared native library binary

Opening the Black Box of Android Packer

Pack Provider	Modified/Replaced File	Comments
ApkProtect	classes.dex	// Modified original classes.dex file
Bangcle	AndroidManifest.xml classes.dex	// Configure to implement Bangcle class // classes.dex replaced by Bangcle
Ijiami	AndroidManifest.xml classes.dex	// Configure to implement Ijiami class // classes.dex replaced by Ijiami

Code Tree of Decompiled classes.dex



Original classes.dex

ApkProtect

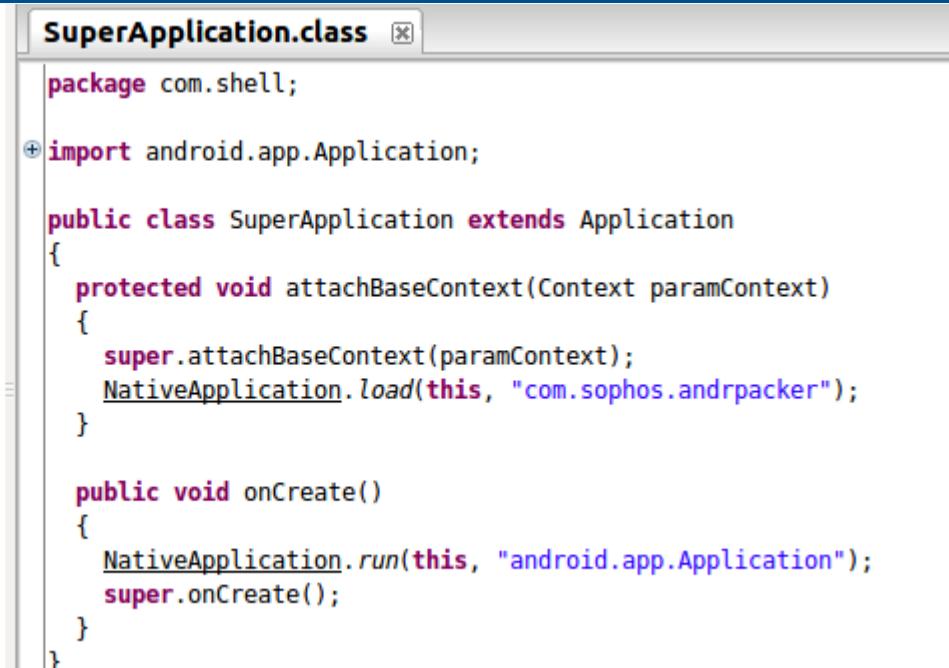
Ijiami

Bangcle

Investigation on Ijiami – entrypoint

Application class – the start point to execute unpacker

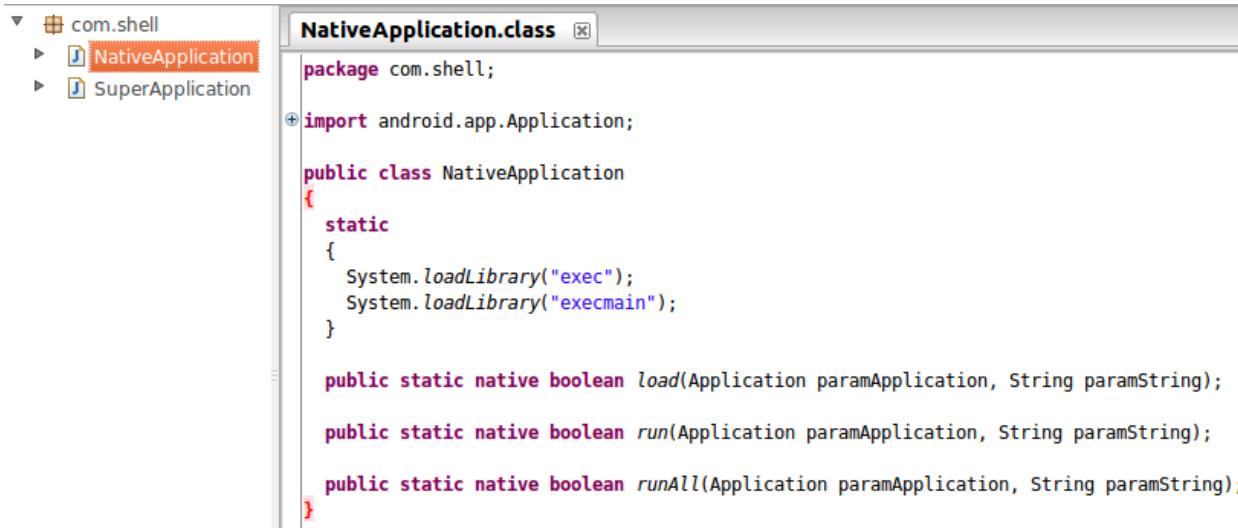
```
<application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/ic_launcher"  
android:label="@string/app_name" android:theme="@style/AppTheme">  
  
<application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/ic_launcher"  
android:label="@string/app_name" android:name="com.shell.SuperApplication"  
android:theme="@style/AppTheme">
```



The screenshot shows an IDE interface with a file tree on the left and a code editor on the right. The file tree shows a package named 'com.shell' containing two classes: 'NativeApplication' and 'SuperApplication'. The 'SuperApplication' class is currently selected and displayed in the code editor.

```
SuperApplication.java  
package com.shell;  
  
import android.app.Application;  
  
public class SuperApplication extends Application  
{  
    protected void attachBaseContext(Context paramContext)  
    {  
        super.attachBaseContext(paramContext);  
        NativeApplication.load(this, "com.sophos.andrpacker");  
    }  
  
    public void onCreate()  
    {  
        NativeApplication.run(this, "android.app.Application");  
        super.onCreate();  
    }  
}
```

Investigation on Ijiami – working process



The screenshot shows a Java code editor with the file `NativeApplication.class` open. The code is as follows:

```
package com.shell;
import android.app.Application;
public class NativeApplication
{
    static
    {
        System.loadLibrary("exec");
        System.loadLibrary("execmain");
    }

    public static native boolean load(Application paramApplication, String paramString);
    public static native boolean run(Application paramApplication, String paramString);
    public static native boolean runAll(Application paramApplication, String paramString);
}
```

- libexec.so
 - Its the main code is `d_deinit`.
 - It verifies the signature and integrity of encrypted data.
 - It decrypts assets/ijiami.dat to original classes.dex.
- libexecmain.so
 - It calls `d_deinit` and `startLoad` and `run` functions in Dalvik layer.

Investigation on Ijiami – dex header modification

Modify original dex header in memory

The modification starts from the beginning of dex file to 0x28 bytes

Additional studies on **BANGCLE**梆梆

Entrypoint of Bangcle source code – ApplicationWrapper class

```
public void onCreate()
{
    super.onCreate();
    if (Util.getCustomClassLoader() == null) {
        Util.runAll(this);
    }
    String str = FirstApplication;
    try
    {
        this.cl = ((DexClassLoader)Util.getCustomClassLoader());
        realApplication = (Application)getClassLoader().loadClass(str).newInstance();
        if (realApplication != null)
        {
            localACall = ACall.getACall();
            localACall.at1(realApplication, getBaseContext());
            localACall.set2(this, realApplication, this.cl, getBaseContext());
        }
    }...
```

Additional studies on **BANGCLE**梆梆

runAll method of Bangcle Util class

```
public static void runAll(Context paramContext)
{
    x86Ctx = paramContext;
    doCheck(paramContext); // checking integrity of classes.jar
    checkUpdate(paramContext);
    try
    {
        File localFile = new File("/data/data/" + paramContext.getPackageName() +
"/.cache/");
        if (!localFile.exists())
        {
            localFile.mkdir();
        }
        checkX86(paramContext); // If it is x86 platform, copy related library binary
        CopyBinaryFile(paramContext); // copy encrypted classes.jar and JNI binary
        createChildProcess(paramContext); // create child processes
        tryDo(paramContext);
        runPkg(paramContext, paramContext.getPackageName()); // call MyClassLoader
        return;
    } ...
```

Additional studies on **BANGCLE**梆梆

- the ACall class of Bangcle deals with binary such as libsecexe.so in the Library layer of Android
- libsecexe.so is responsible for JNI call

```
public native void a1(byte[] paramArrayOfByte1, byte[] paramArrayOfByte2);  
  
public native void at1(Application paramApplication, Context paramContext);  
  
public native void at2(Application paramApplication, Context paramContext);  
  
public native void c1(Object paramObject1, Object paramObject2);  
  
public native void c2(Object paramObject1, Object paramObject2);  
  
public native Object c3(Object paramObject1, Object paramObject2);  
  
public native void r1(byte[] paramArrayOfByte1, byte[] paramArrayOfByte2);  
  
public native void r2(byte[] paramArrayOfByte1, byte[] paramArrayOfByte2, byte[] paramArrayOfByte3);  
  
public native ClassLoader rc1(Context paramContext);  
  
public native void s1(Object paramObject1, Object paramObject2, Object paramObject3);  
  
public native Object set1(Activity paramActivity, ClassLoader paramClassLoader);  
  
public native Object set2(Application paramApplication1, Application paramApplication2, ClassLoader paramClassLoader);
```

pA226AD0639E094643D446D114B40A4F7	0001072C
p14285A16A9AD09C58C6229A0216C2BCE	00009E6C
pFB0F628D4A0CEDB94B22B8AF32C6449	0000E1C0
pFFB607FCF6C8C78DF1B93B14618C1170	00021E60
p48661E70C9925A280F22F90CE1DD9FBC	0000A100
p6543834C664025CDB9CC8865EA4F5D21	00008744
pBAE09FC1D43B26EF272F4502C9B9A761	00021E64
p614EBEA527F7CFE77711182EACCBC3CE	00021E68
p2D656B85C816001EDC4DBA95AD2B1451	0000BBB4
p9E0BA5F141B271A7182A3D7E36F3B98C	00021B00
p59E15566C42CB17277A9BC11BD48E66D	00021E6C
p6681D68CA8B7E8F086ECE19A06ED13D0	0000B238
pA3E4F5DB10866DA44836DD6A227D7FE5	000216EC
p261362038FFCD16031F67682A01C2B16	0000F9A0
p92B745BE458923D2EFA4F4E95A0A5A51	0000E0A4
p6C5C888FDD1835CEC733F97BBA04B9B2	00021C00
p654E3EE0BCC4136DD6A880AF954A8AC0	0000E118
p403FB1BE0452ED04B365494693196D2F	00021AFC
pF0F81B20C6DEDD307E414D3004F853B6	00021C04
pC0B5AC27AB6DC40D0C0A378EC7ED634B	0000FC5C
p5FD2555384803B7DEE6F72B840DCFB	00005304
JNI_OnLoad	00007278

Additional studies on **BANGCLE梆梆**

Mutually tracing in three Bangcle processes
to block analysis from debugging tools

```
u0_a46    1552  57   203288 22432 ffffffff 40063ebc S a.hello
u0_a46    1568  1552  b6944p :1590 l 0fffff5fd4003ad5c S a.hello
u0_a46    1570  1568  2076   372   c00c2e10 40037d50 S a.hello
```

```
dmesg | grep -i ptrace
<4>anti-ptrace kernel module loaded with pid=[1398]
<4>PTTRACE: pid=[1568] uid=[10046], [16,1552, (null), (null) ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [16,1570, (null), (null) ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [07,1570, (null), (null) ==> 0 ]
<4>PTTRACE: pid=[1570] uid=[10046], [07,1568, (null), (null) ==> -3 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a8446f0,12345678 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a8446f4,563ffbcf ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a8446f8,5c745a41 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a8446fc,c5f094b7 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844700,4495deba ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844704,f689a279 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844708,55e102c3 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a84470c,a3b885d5 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844710,6b6d5841 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844714,f689a279 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844718,55e102c3 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a84471c,a3b885d5 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [05,1552,4a844720,6b6d5841 ==> 0 ]
<4>PTTRACE: pid=[1568] uid=[10046], [07,1552, (null), (null) ==> 0 ]
```

Technology Review of Bangcle and Ijiami

- Anti-Temper – checks the integrity check of encrypted dex.
- Anti-Decompiler – decrypts the encrypted dex in memory and employs MyClassLoader to load decrypted original dex in runtime.
- Anti-Runtime injection – It is impossible to establish relationship between ACall class with libsecexe.so due to the encryption.
- Anti-Debug – Both packers use techniques to block analysis from debugging tools

PART 2

Facing Challenges - Ineffective reverse engineering (RE) tools

- IDA Pro
- Dex2jar
- Apktool
- BakSmali/Smali
- JEB

The screenshot displays four windows from different tools, likely showing the same or similar Java code for an application named 'Bangcle'.

- Bangcle_com.sophos.andrpacker-dex2jar.jar:** A file browser view showing a package structure under 'com'. The 'secapk.wrapper' package contains classes: ACall, ApplicationWrapper, FirstApplication, MyClassLoader, and Util. The 'ACall' class is selected and shown in its source code.
- Ijiami_com.sophos.andrpacker-dex2jar.jar:** A file browser view showing a package structure under 'com.shell'. It contains NativeApplication and SuperApplication classes. The NativeApplication class is selected and shown in its source code.
- SuperApplication.class:** A code editor window showing the source code for the SuperApplication class. It imports android.app.Application and defines a static method System.loadLibrary("exec");
- NativeApplication.class:** A code editor window showing the source code for the NativeApplication class. It imports android.app.Application and defines a static method System.loadLibrary("exec");
- ijiami.dat:** A hex editor window showing the raw binary data of the 'ijiami' file.
- bangcle_classes.jar:** A hex editor window showing the raw binary data of the 'bangcle_classes' jar file.

Facing Challenges - Failure of dynamic analysis systems

基本信息 其他行为 权限列表 启动方式

基本信息

文件名称： Ijiami_com.sophos.andrpacker.apk
MD5： ac8a2656fb865a854bfc906cec744947
Sha-1： f8435c1485963994b778d28c36ad34613369f26b
文件大小： 543KB
创建日期： 2014-04-26 19:45:33
应用名称： AndrPacker
证书信息： /C=Unknown/ST=Unknown/L=Unknown/O=Unknown/OU=Unknown/CN=Unkn
owr文件包名： com.sophos.andrpacker
版本信息： 1.0

其他行为监控

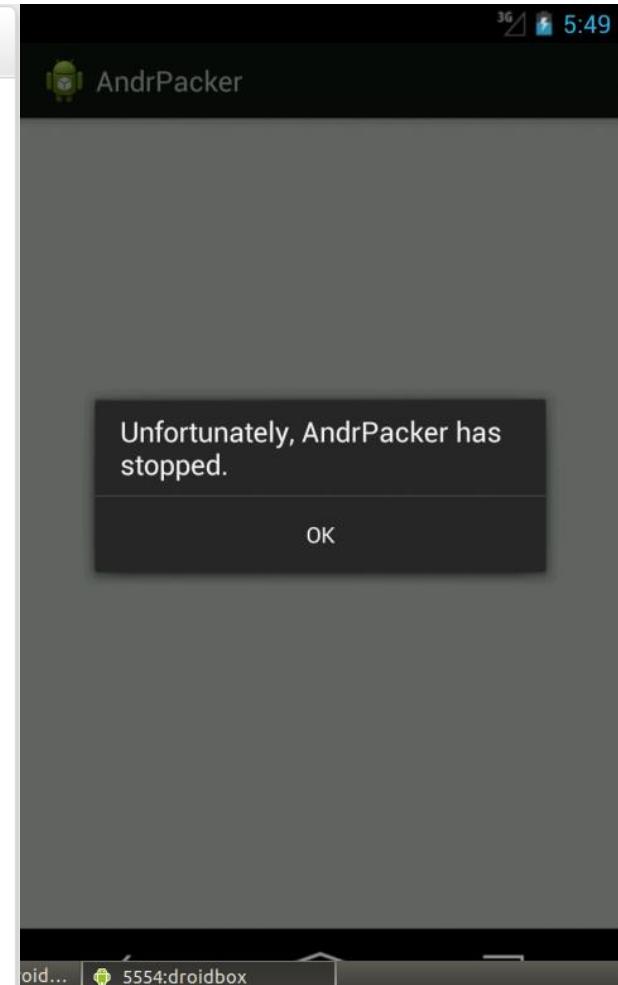
行为描述： 程序加固
附加信息： 爱加密 <http://www.jjiami.cn/>

权限列表

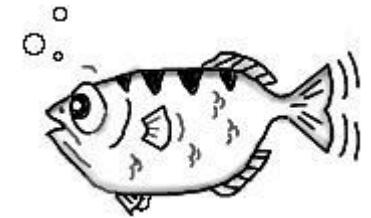
• 高危 • 危险 • 普通

监控接收短信	已使用	android.permission.RECEIVE_SMS
接收开机启动广播	已使用	android.permission.RECEIVE_BOOT_COMPLETED

启动方式



Facing Challenges - Runtime Anti-Debug



- gdb, gcore, memory searching, dd
- Anti-ptrace technique to block analysis from debugging tools

Facing Challenges - Difficult to detect by security solutions

- Certificate
- Package name
- AndroidManifest.xml
-



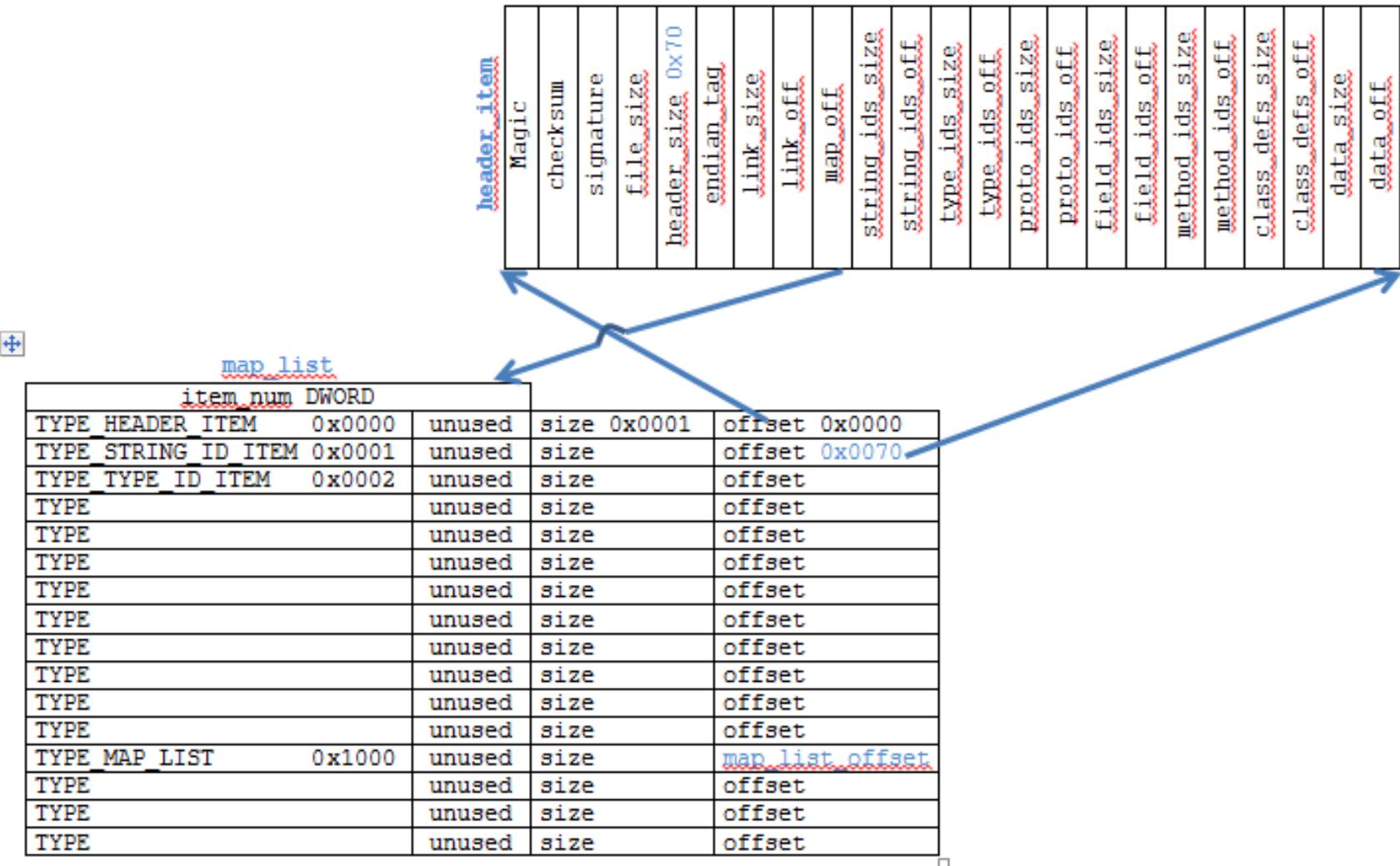
Building Solutions – Acquire Memory

1. Initialize an Android build environment including path and required package on either Linux or OSX system.
2. Download the Android SDK and NDK.
3. Download the Android Kernel Source Code.
4. Cross Compile the Kernel.
5. Create AVD then Emulate the Custom Kernel with the AVD.
6. Download and Cross Compile LiME.
7. Load LiME on the Android Device/Emulator.
8. Acquire Memory.

Building Solutions – Memory forensics with Volatility plugins

- linux_pslist – gathers active tasks by walking the task_struct.
- linux_proc_maps – gathers process maps for Linux.
- linux_dump_map – writes selected process memory mappings to disk.

Building Solutions – Memory forensics with Volatility plugins



Building Solutions – Memory forensics with Volatility plugins apk_packer_find_dex

```
signatures = {
    'map_header' : 'rule map_header { \
        strings: \
            $hex = {00 00 ?? ?? 01 00 00 00 00 00 00 00 00 01 00 ?? ?? ?? ?? ?? ?? ?? ?? 70
00 00 00 02 00} \
        condition: $hex }'
}

class apk_packer_find_dex(linux_common.AbstractLinuxCommand):
    """Gather information about the dex Dump in Memory running in the
    system"""

    def __init__(self, config, *args, **kwargs):
        linux_common.AbstractLinuxCommand.__init__(self, config, *args, **kwargs)
        self._config.add_option('PID', short_option='p', default=None,
            help='Operate on a specific Android application Process ID',
            action='store', type='str')
```

```
def calculate(self):
    """ Required: Runs YARA search to find hits """
    rules = yara.compile(sources = signatures)

    proc_maps = linux_proc_maps.linux_proc_maps(self._config).calculate()
    for task, vma in proc_maps:
        if not vma.vm_file:
            ...
            proc_as = task.get_process_address_space()
            maxlen = vma.vm_end - vma.vm_start
            data = proc_as.zread(vma.vm_start, maxlen - 1)

            if data:
                for match in rules.match(data = data):
                    for moffset, _name, _value in match.strings:
                        (usize,) = struct.unpack('I', data[moffset - 4 : moffset])

                        i = 0
                        offset = moffset
                        while i < usize:

                            (maptype,) = struct.unpack('H', data[offset: offset+2])
                            (mapoffset,) = struct.unpack('I', data[offset+8: offset+12])

                            if maptype == 0x1000:
                                yield task, vma, moffset - 4 - mapoffset, moffset
                                break
                            i += 1
                            offset += 12
```

Building Solutions – Memory forensics with Volatility plugins apk_packer_find_dex

```
def render_text(self, outfd, data):
    self.table_header(outfd, [ ("Task", "10"),
        ("VM Start", "[addrpad]"),
        ("VM End", "[addrpad]"),
        ("Dex Offset", "[addr]"),
        ("Map Offset", "[addr]") ])
    for (task, vma, offset, moffset) in data:
        self.table_row(outfd, task.pid, vma.vm_start,
vma.vm_end, offset, moffset - 4)
```

Demonstration – Memory Acquisition

```
emulator -wipe-data -avd myavd -kernel ~/android-
kernel/arch/arm/boot/zImage -show-kernel -verbose

adb push ~/lime-forensics/src/lime-goldfish.ko /sdcard/lime.ko

adb shell

root@android:/ #

insmod /sdcard/lime.ko "path=/sdcard/lime.dump format=lime"

root@android:/ #

ls -al /sdcard/lime.dump

fls ~/.android/avd/myavd.avd/sdcard.img

icat ~/.android/avd/myavd.avd/sdcard.img 27 > ~/lime.dmp
```

Demonstration – Bangcle

1. Find PID:

```
python ~/android-volatility/vol.py --profile=LinuxGolfish-  
2_6_29ARM -f Bangcle_lime.dmp linux_pslist
```

0xf2f90c00	phos.andrpacker	876	10046	10046	0x263b8000	2014-04-14 02:40:37 UTC+0
0xe9159c00	com.sophos.andr	891	10046	10046	0x2add0000	2014-04-14 02:40:38 UTC+0
0xe631f000	com.sophos.andr	893	10046	10046	0x2af14000	2014-04-14 02:40:38 UTC+0

2. Find original dex in memory:

```
python ~/android-volatility/vol.py --profile=LinuxGolfish-  
2_6_29ARM -f Bangcle_lime.dmp apk_packer_find_dex -p876
```

Task	VM Start	VM End	Dex Offset	Map Offset
876	0x4c10d000	0x4c1a4000	0x28	0x8ffc8

Demonstration – Bangcle

3. Dump memory map:

```
python ~/android-volatility/vol.py --profile=LinuxGolfish-  
2_6_29ARM -f Bangcle_lime.dmp linux_dump_map -p876 -s  
0x4c10d000 --dump-dir ~/Downloads/
```

```
Volatility Foundation Volatility Framework 2.3.1  
Task      VM Start   VM End       Length Path  
-----  
 876 0x4c10d000 0x4c1a4000    0x97000 /home/labrat/Downloads/task.876.0x4c10d000.vma
```

4. Truncate to get ‘original’ dex:



The screenshot shows the JD-GUI interface with the title bar "task.876.0x4c10d000-dex2jar.jar". On the left is a tree view of the class hierarchy:

- android.support.v4
- com.sophos.andrpacker
- neo
 - proxy
 - ContainerFactory
 - DistributeReceiver
 - FastService
 - skeleton
 - base
 - Actions
 - Coder
 - Configs
 - Configurable
 - Containable
 - NetStream
 - Plugable
 - Plugin
 - SafeConfigs
 - Storage

On the right, the code editor displays the content of the `FastService.class` file:

```
package neo.proxy;  
  
import android.app.IntentService;  
  
public class FastService extends IntentService  
{  
    static final String TAG = "FastService";  
    private Containable container;  
    private Map<String, Long> histories = new HashMap();  
  
    public FastService()  
    {  
        super("FastService");  
    }  
  
    public void onDestroy()  
    {  
        if (this.container != null)  
            this.container.clean();  
        ContainerFactory.clearCachedContainer();  
        super.onDestroy();  
    }  
}
```

Demonstration – Ijiami

Patch DEX_FILE_MAGIC header

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDE
0000	C1	11	05	00	97	1C	05	00	F0	1F	05	00	BA	17	05	00xV4..
0010	14	07	05	00	D2	10	05	00	D7	13	05	00	E2	0F	05	00p
0020	5E	0B	05	00	D9	10	05	00	78	56	34	12	00	00	00	004`...]
0040	00	00	00	00	04	BC	01	00	F1	17	00	00	70	00	00	00	X.....t.....
0050	61	03	00	00	34	60	00	00	5D	04	00	00	B8	6D	00	001....1....
0060	58	05	00	00	14	A2	00	00	F9	14	00	00	D4	CC	00	001....1....
0070	0B	02	00	00	9C	74	01	00	1C	F6	08	00	04	BC	01	001....1....
0080	80	6C	06	00	82	6C	06	00	85	6C	06	00	88	6C	06	001....1....
0090	8C	6C	06	00	92	6C	06	00	97	6C	06	00	AE	6C	06	001....1....
00A0	D6	6C	06	00	E6	6C	06	00	09	6D	06	00	23	6D	06	001....m..#
00B0	40	6D	06	00	61	6D	06	00	79	6D	06	00	A0	6D	06	00	@m..am..ym..
00C0	C8	6D	06	00	E9	6D	06	00	F1	6D	06	00	OE	6E	06	00	.m...m...m...
00D0	31	6E	06	00	4E	6E	06	00	5C	6E	06	00	6B	6E	06	00	1n..Nn..\n..ki
00E0	79	6E	06	00	87	6E	06	00	A8	6E	06	00	B6	6E	06	00	yn...n...n...
00F0	C9	6E	06	00	D8	6E	06	00	E6	6E	06	00	F9	6E	06	00	.n...n...n...
0100	11	6F	06	00	2C	6F	06	00	38	6F	06	00	4E	6F	06	00	.o.,o..8o..Ne
0110	52	6F	06	00	56	6F	06	00	A0	6F	06	00	AA	6F	06	00	Ro..Vo...o...
0120	AE	6F	06	00	B3	6F	06	00	B9	6F	06	00	C9	6F	06	00	.o...o...o...
	D6	6F	06	00	EE	6F	06	00	F7	6F	06	00	08	70	06	00	.o...o...o...

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDE
0000	64	65	78	0A	30	33	35	00	F0	1F	05	00	BA	17	05	00	dex.035.....
0010	14	07	05	00	D2	10	05	00	D7	13	05	00	E2	0F	05	00xV4..
0020	5E	0B	05	00	D9	10	05	00	78	56	34	12	00	00	00	00p
0030	00	00	00	00	04	BC	01	00	F1	17	00	00	70	00	00	004`...]
0040	61	03	00	00	34	60	00	00	5D	04	00	00	B8	6D	00	00	X.....t.....
0050	58	05	00	00	14	A2	00	00	F9	14	00	00	D4	CC	00	001....1....
0060	0B	02	00	00	9C	74	01	00	1C	F6	08	00	04	BC	01	001....1....
0070	80	6C	06	00	82	6C	06	00	85	6C	06	00	88	6C	06	001....1....
0080	8C	6C	06	00	92	6C	06	00	97	6C	06	00	AE	6C	06	001....1....
0090	D6	6C	06	00	E6	6C	06	00	09	6D	06	00	23	6D	06	001....m..#
00A0	40	6D	06	00	61	6D	06	00	79	6D	06	00	A0	6D	06	00	@m..am..ym..
00B0	C8	6D	06	00	E9	6D	06	00	F1	6D	06	00	OE	6E	06	00	.m...m...m...
00C0	31	6E	06	00	4E	6E	06	00	5C	6E	06	00	6B	6E	06	00	1n..Nn..\n..ki
00D0	79	6E	06	00	87	6E	06	00	A8	6E	06	00	B6	6E	06	00	yn...n...n...
00E0	C9	6E	06	00	D8	6E	06	00	E6	6E	06	00	F9	6E	06	00	.n...n...n...
00F0	11	6F	06	00	2C	6F	06	00	38	6F	06	00	4E	6F	06	00	.o.,o..8o..Ne
0100	52	6F	06	00	56	6F	06	00	A0	6F	06	00	AA	6F	06	00	Ro..Vo...o...
0110	AE	6F	06	00	B3	6F	06	00	B9	6F	06	00	C9	6F	06	00	.o...o...o...
0120	D6	6F	06	00	EE	6F	06	00	F7	6F	06	00	08	70	06	00	.o...o...o...

Detection Solution

- AndroidManifest.xml
- Classname in Bangcle
- Size of encrypted payload
- Resource files
- resources.arsc
-

Summary

- Popular Android packing service – Ijiami and Bangle
- Explosive growth of packed malware
- Anti-Temper, Anti-Decompiler, Anti-Runtime injection and Anti-Debug
- Challenges to security threat researchers
- Solutions Memory forensics with Volatility plugins
- Demonstration

The Future

Obfuscation + Packer

Q&A