Exploring state-of-art packers&obfuscators on ART and How to defeat it

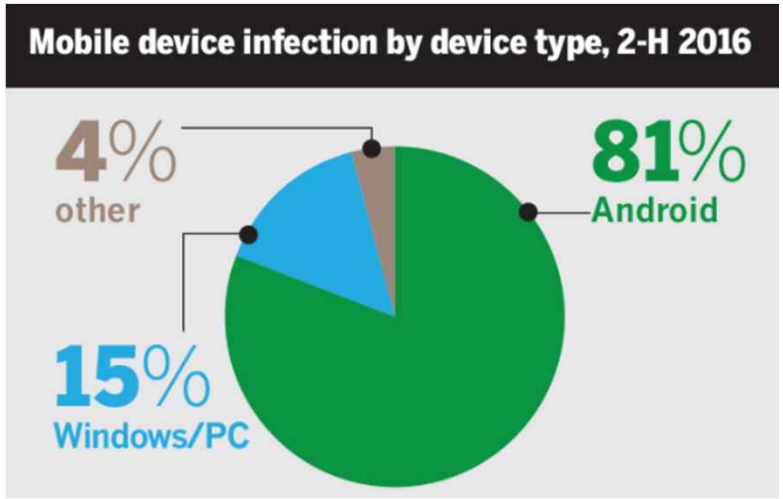
2017. 7. 8

박영웅

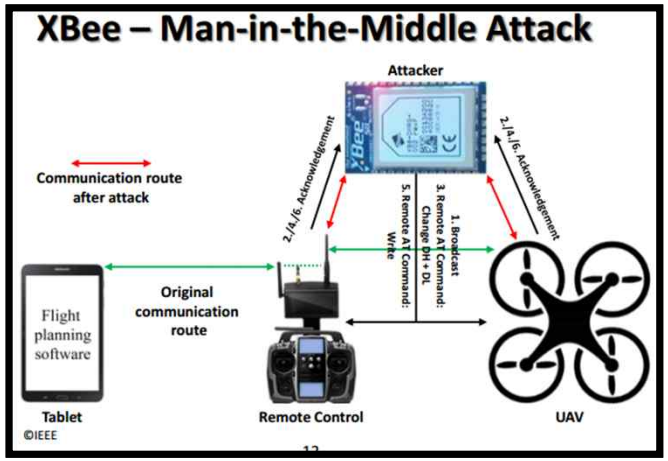
Introduction

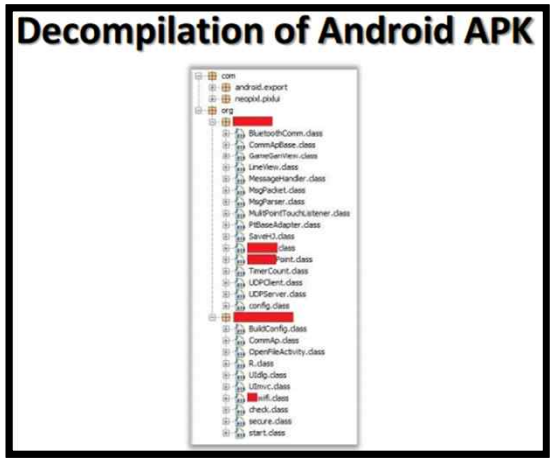
Dvmap: the first Android malware with code injection

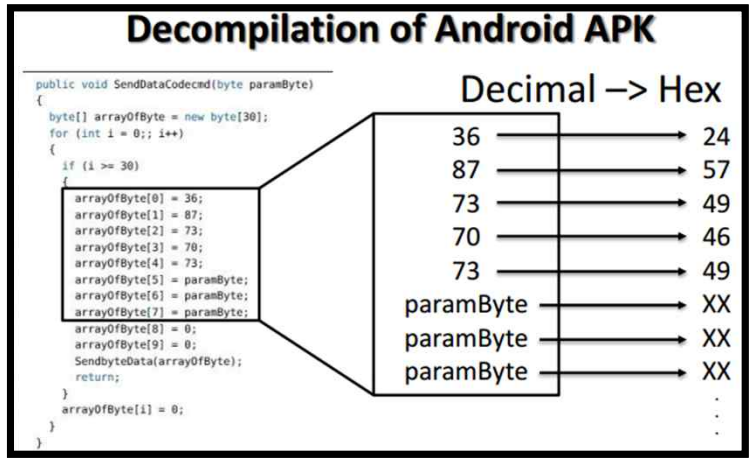
JUDY MALWARE: 36 MILLION ANDROID SMARTPHONES COULD BE INFECTED



- “Hacking a Professional Drone”, Blackhat ASIA 2016







Introduction

- Repackaged app – *(BrainTest, Judy, …)*
- Legimate apps carrying malicious components injected by attackers

1: Packing

Packed Android applications have been around for a long time, but a more recent trend we've observed is the increasing prevalence of Android malware leveraging packing technology. In the past nine months, we've seen the ratio of packer use in malware that our customers encounter increase from 10 percent to 25 percent.

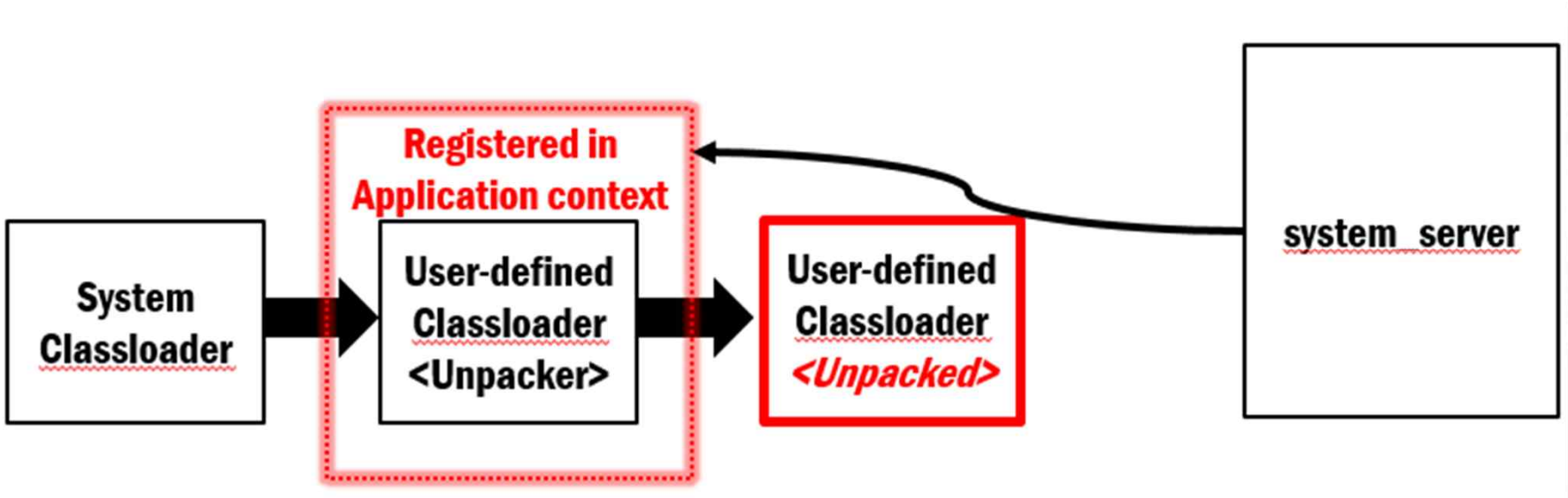
Packer

- Packing mechanism based on Dynamic code loading
  - It can load code ***in file*** *or* ***on memory*** dynamically
  - Android platform provides following interfaces only for Java layer to load .dex file dynamically
    - Documented interfaces: *DexClassLoader, PathClassLoader, DexFile*
    - Undocumented interfaces:
      - ***DexFile.java:***
        - openDexFile(byte[] fileContents)
        - openDexFile(String sourceName, String outputName, int flags)
      - ***dalvik_system_DexFile.cpp:***
        - Dalvik_dalvik_system_DexFile_openDexFile(const u4* args, Jvalue* pResult)
        - Dalvik_dalvik_system_DexFile_openDexFile_bytearray

- Packing mechanism based on Memory patch
  - It modifies <application> tag in AndroidManifest.xml to be executed firstly

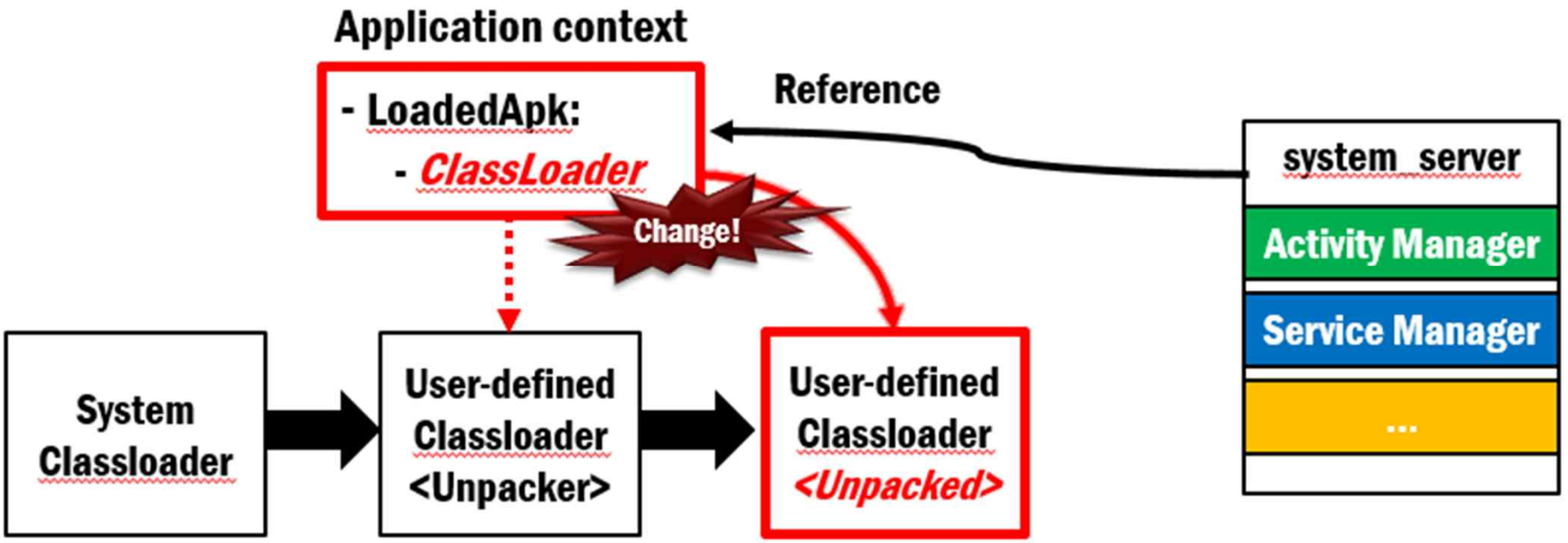
Packer

- Classloader problem by unpacking process



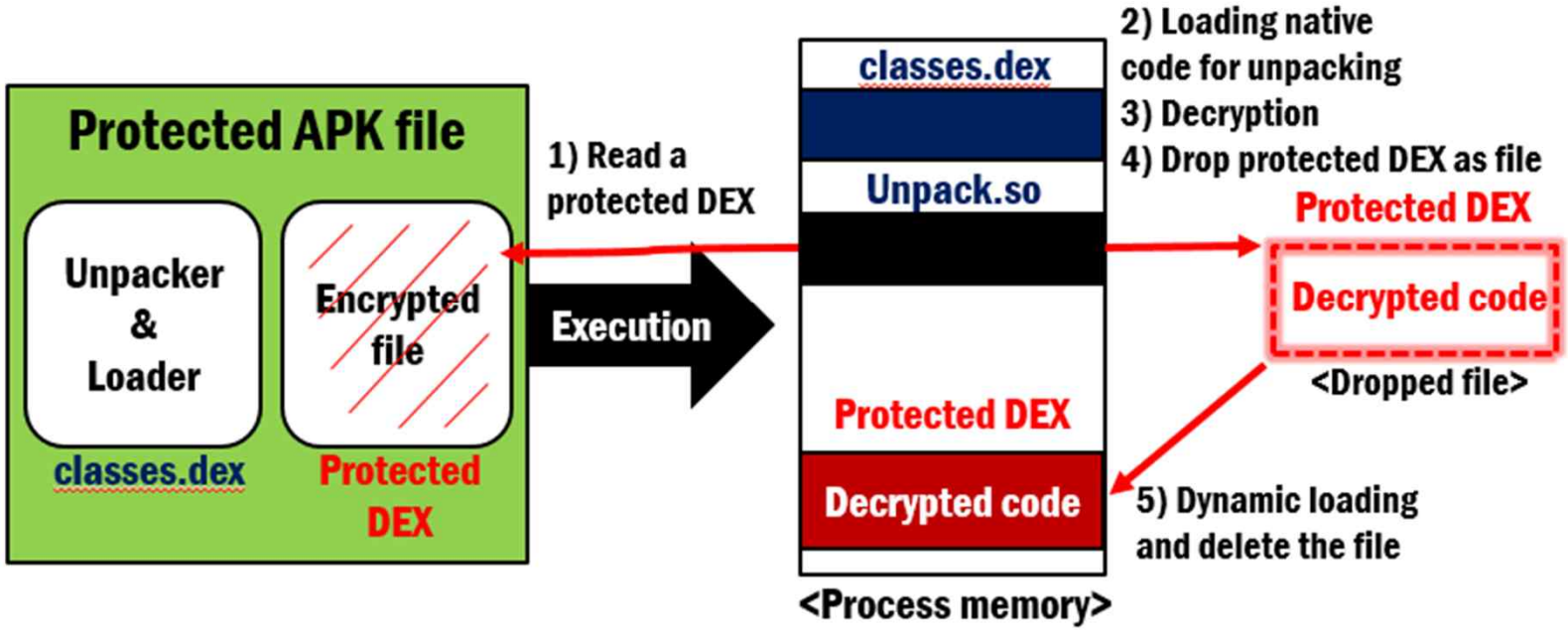
Packer

- Almost packer needs this step



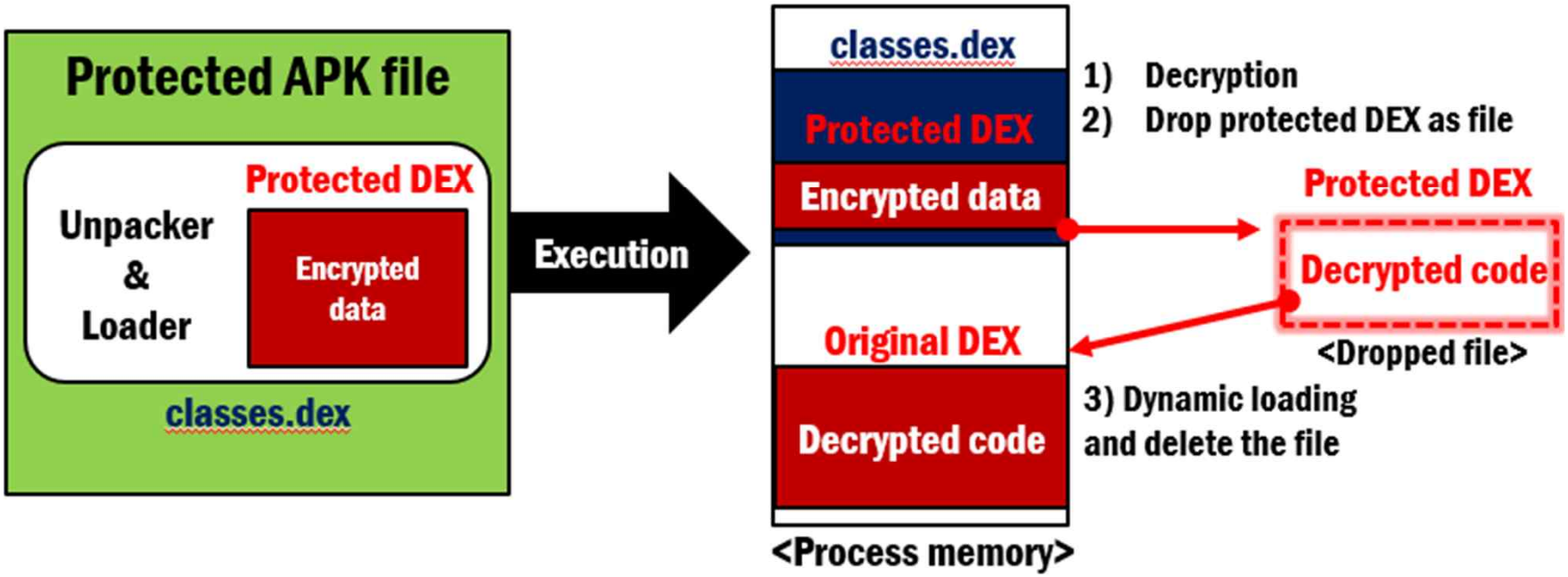
Packer

- Basic type

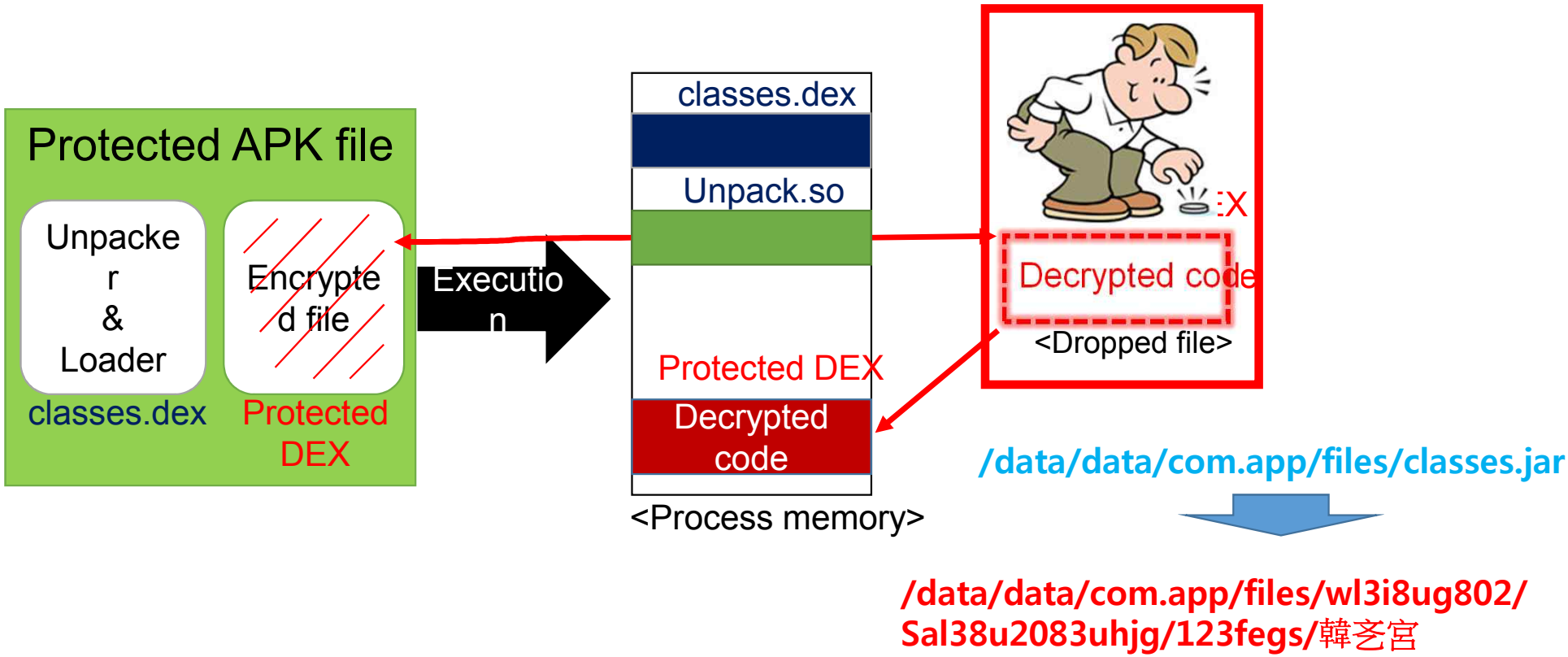


Packer

- Dynamic code loading (***in file***) – DexGuard
  - Protected DEX is in unpacking dex file as array

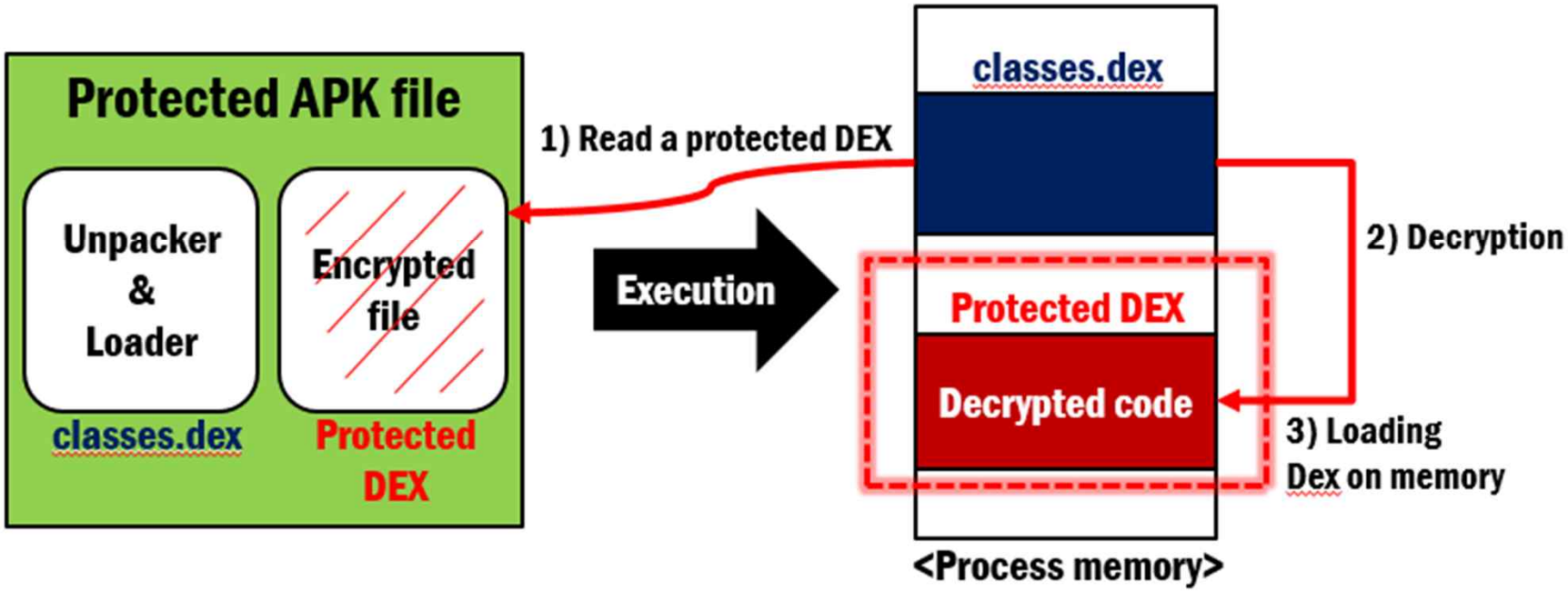


Packer


Protected APK file
Unpacke
r
&
Loader
classes.dex
Encrypte
d file
Protected
DEX
Executio
n
classes.dex
Unpack.so
Protected DEX
Decrypted
code
<Process memory>
Decrypted code
<Dropped file>
/data/data/com.app/files/classes.jar
/data/data/com.app/files/wl3i8ug802/
Sal38u2083uhjg/123fegs/韓玄宮

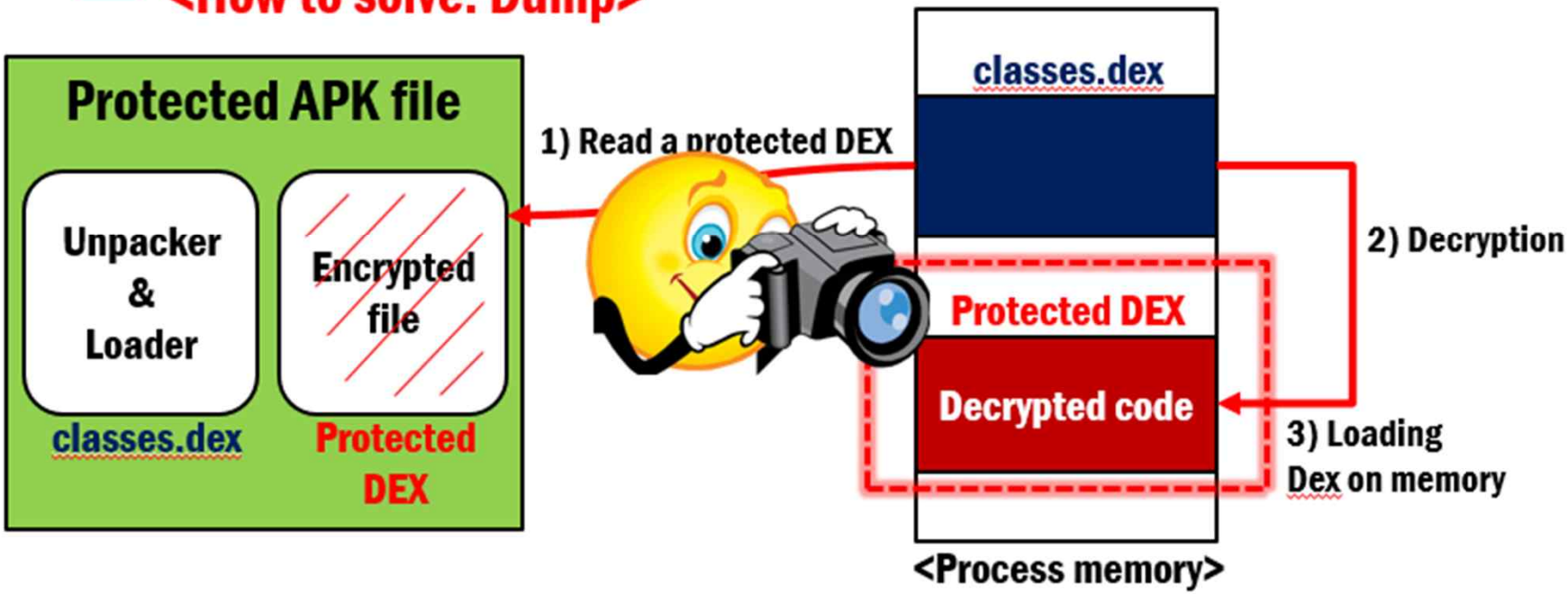
Packer

- Dynamic code loading (***on memory***) - Ijiami

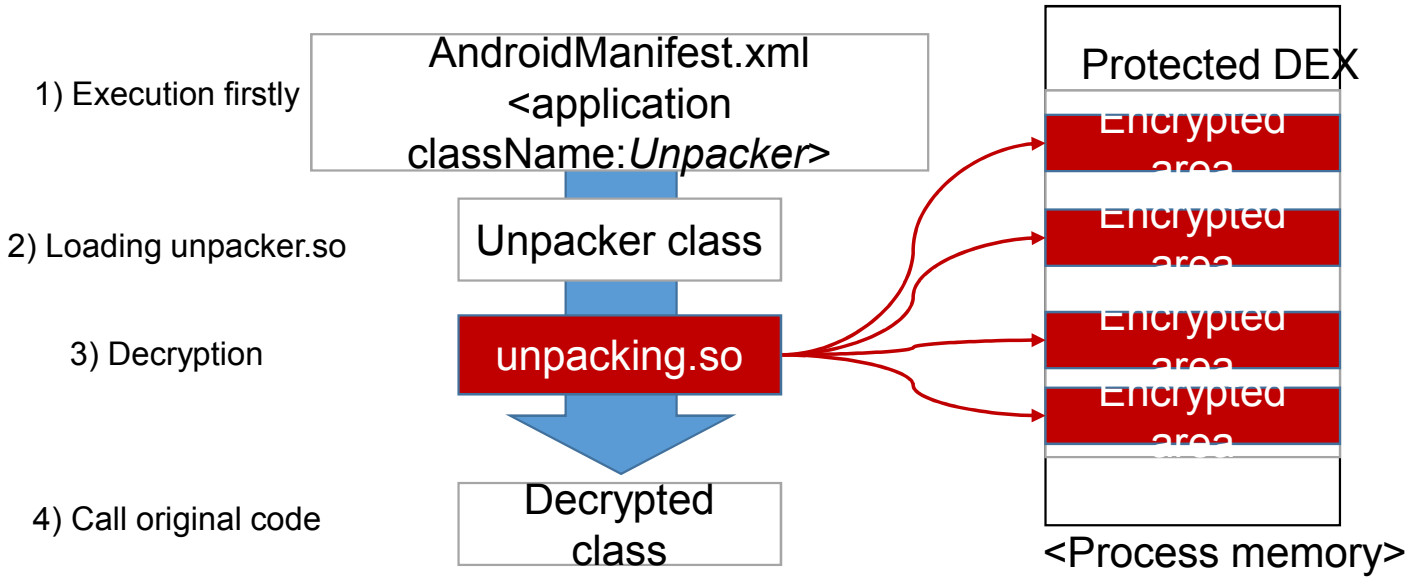


Packer

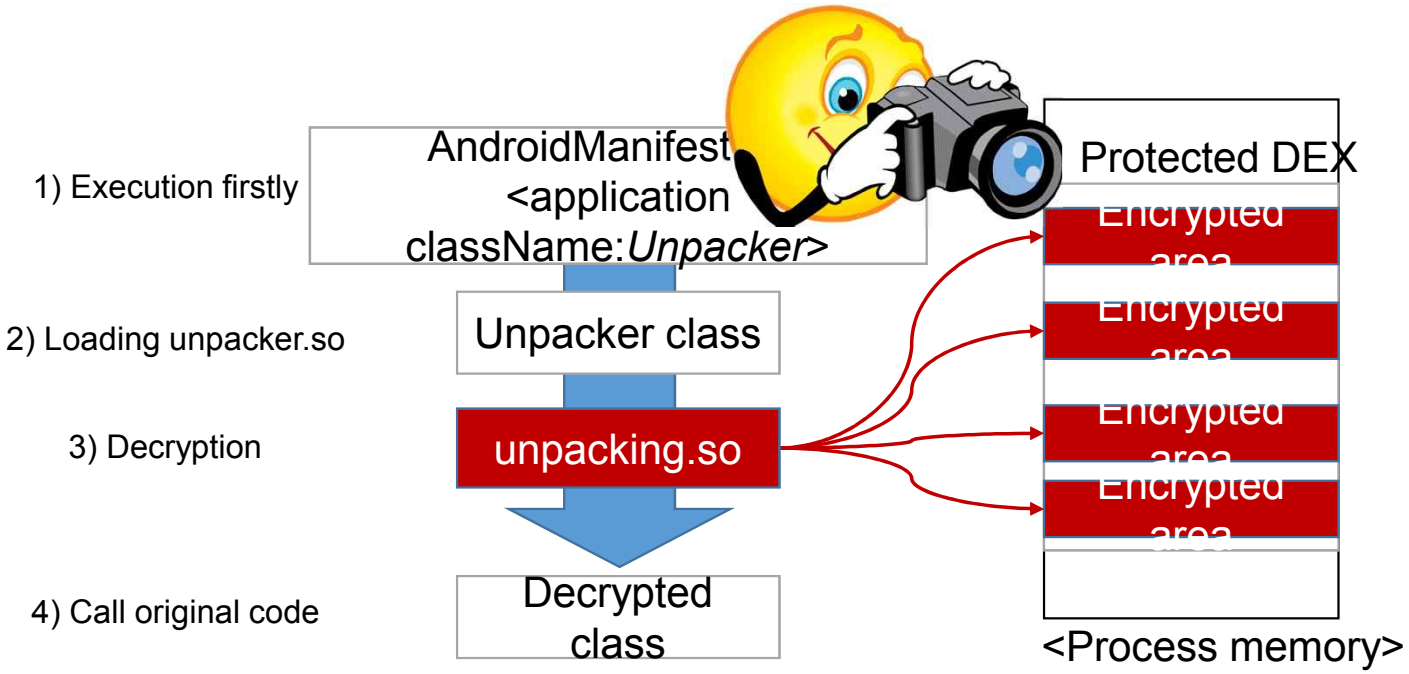
- Dynamic code loading (***on memory***) – Ijiami
  - **<How to solve: Dump>**



Packer

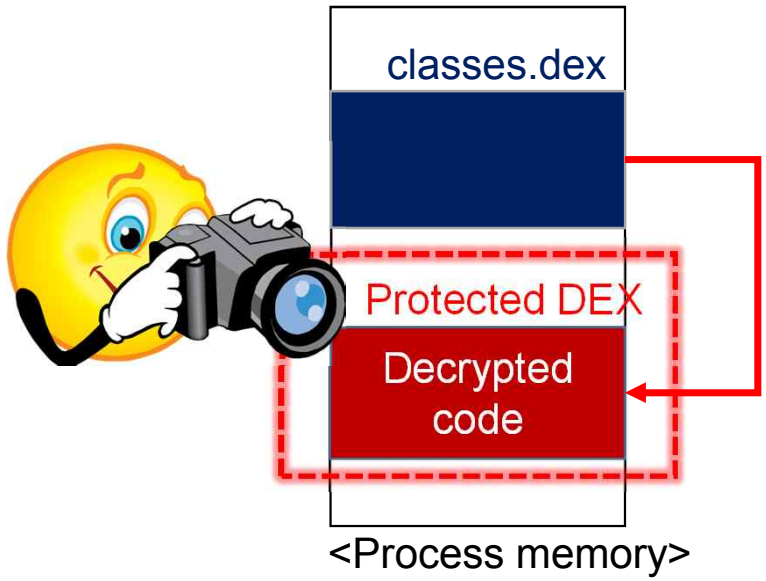

1) Execution firstly
AndroidManifest.xml
<application
className:*Unpacker*>
2) Loading unpacker.so
Unpacker class
3) Decryption
unpacking.so
4) Call original code
Decrypted
class
Protected DEX
Encrypted
area
Encrypted
area
Encrypted
area
Encrypted
area
<Process memory>

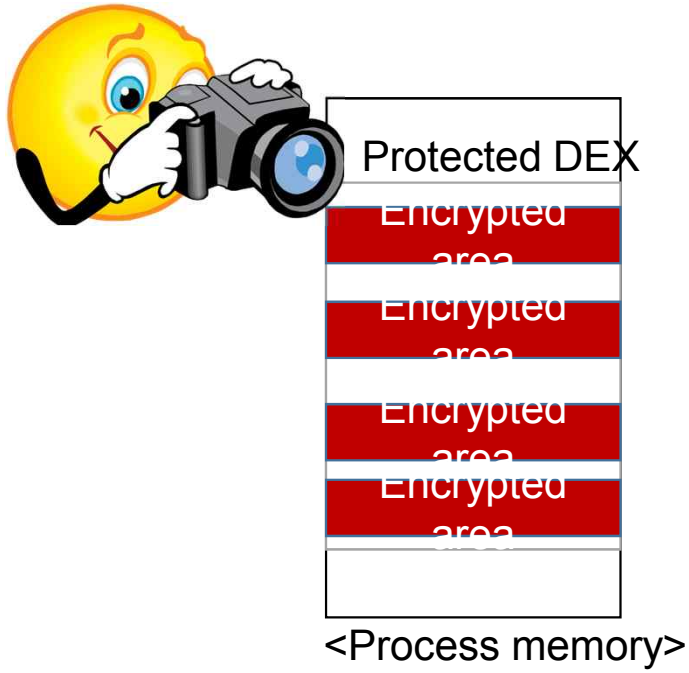
Packer


1) Execution firstly
AndroidManifest
<application
className:*Unpacker*>
2) Loading unpacker.so
Unpacker class
3) Decryption
unpacking.so
4) Call original code
Decrypted
class
Protected DEX
Encrypted
area
Encrypted
area
Encrypted
area
Encrypted
area
<Process memory>

Packer

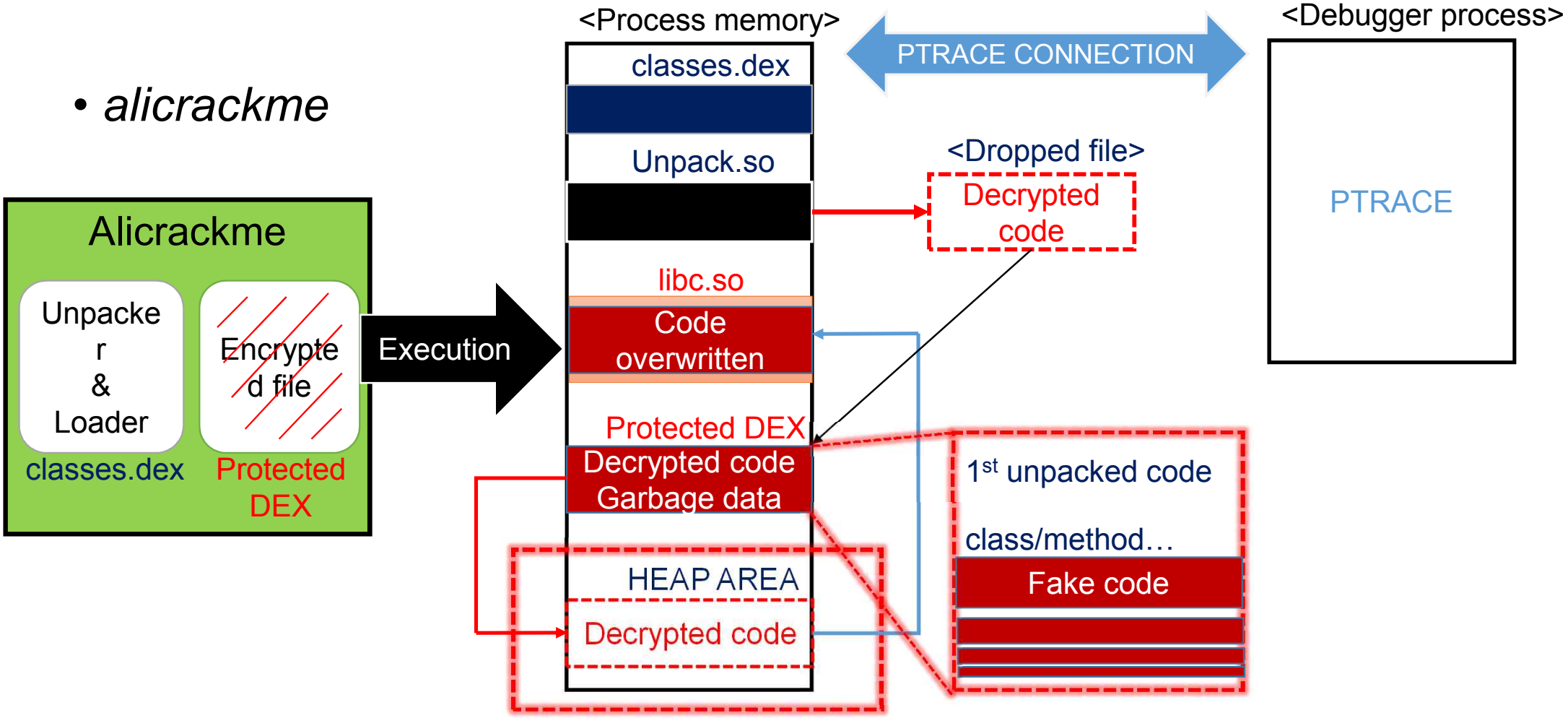
Main unpacking technique: Dump



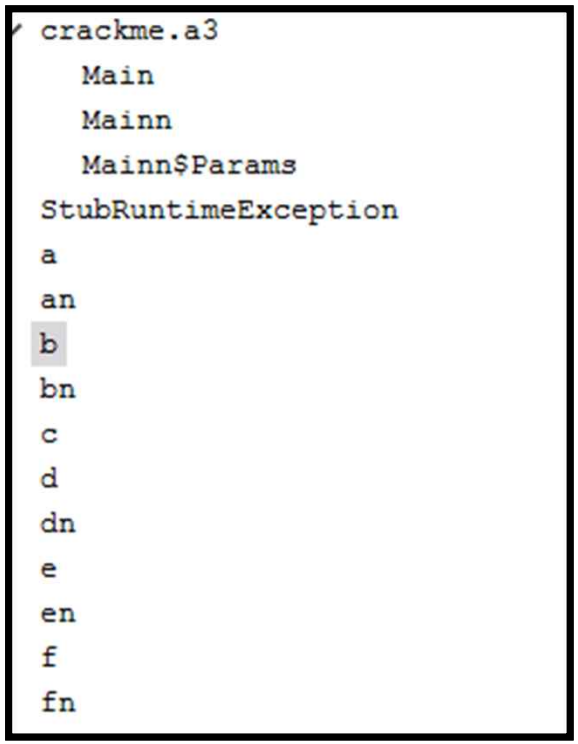


Advanced Packer

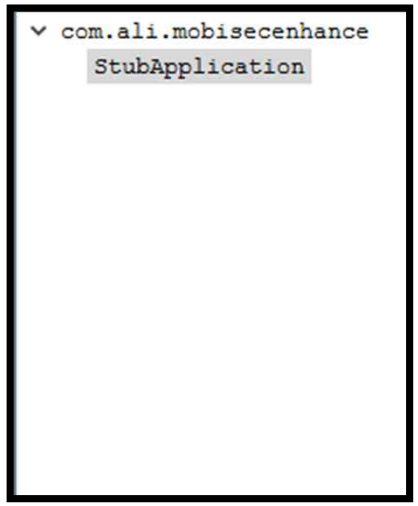
- *alicrackme*



Advanced Packer

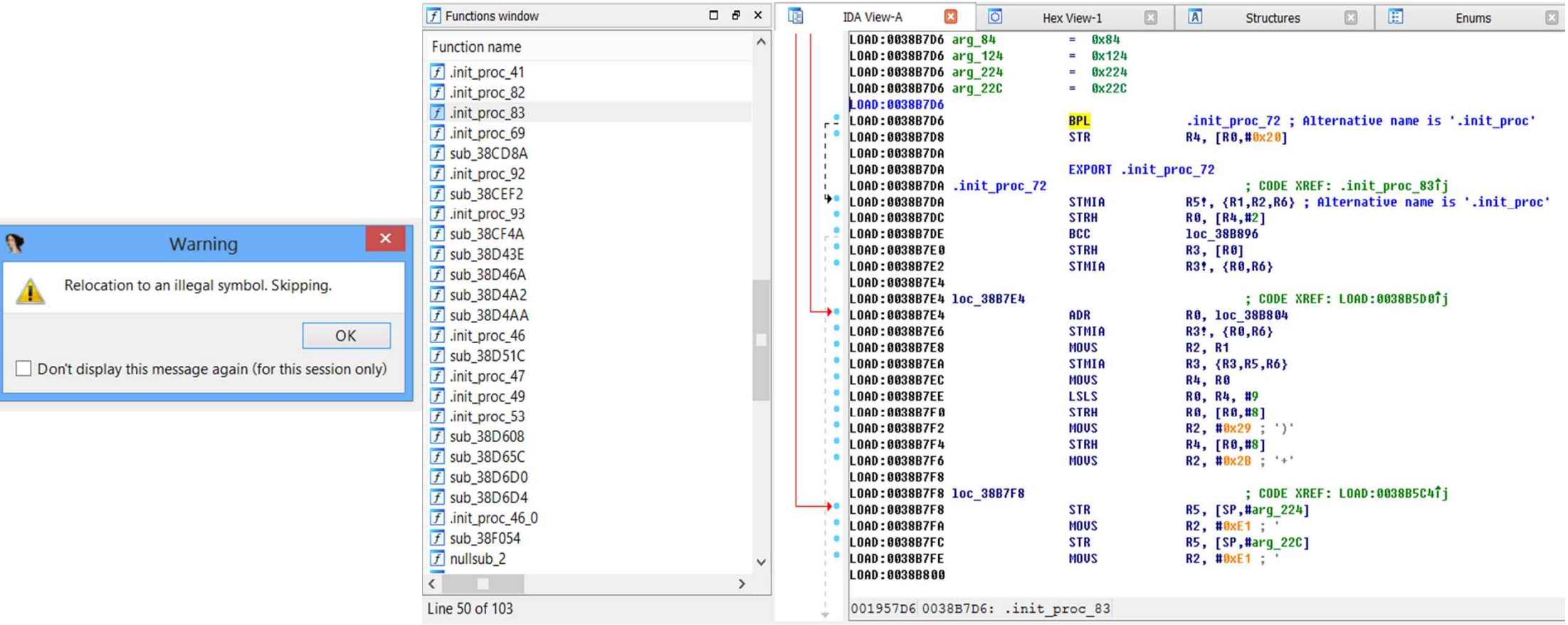



```
import android.os.Handler;
import java.util.TimerTask;

class b extends TimerTask {
    static {
        throw new RuntimeException();
    }

    b(a arg2, Handler arg3, String arg4) {
        throw new RuntimeException();
    }

    public void run() {
        throw new RuntimeException();
    }
}
```

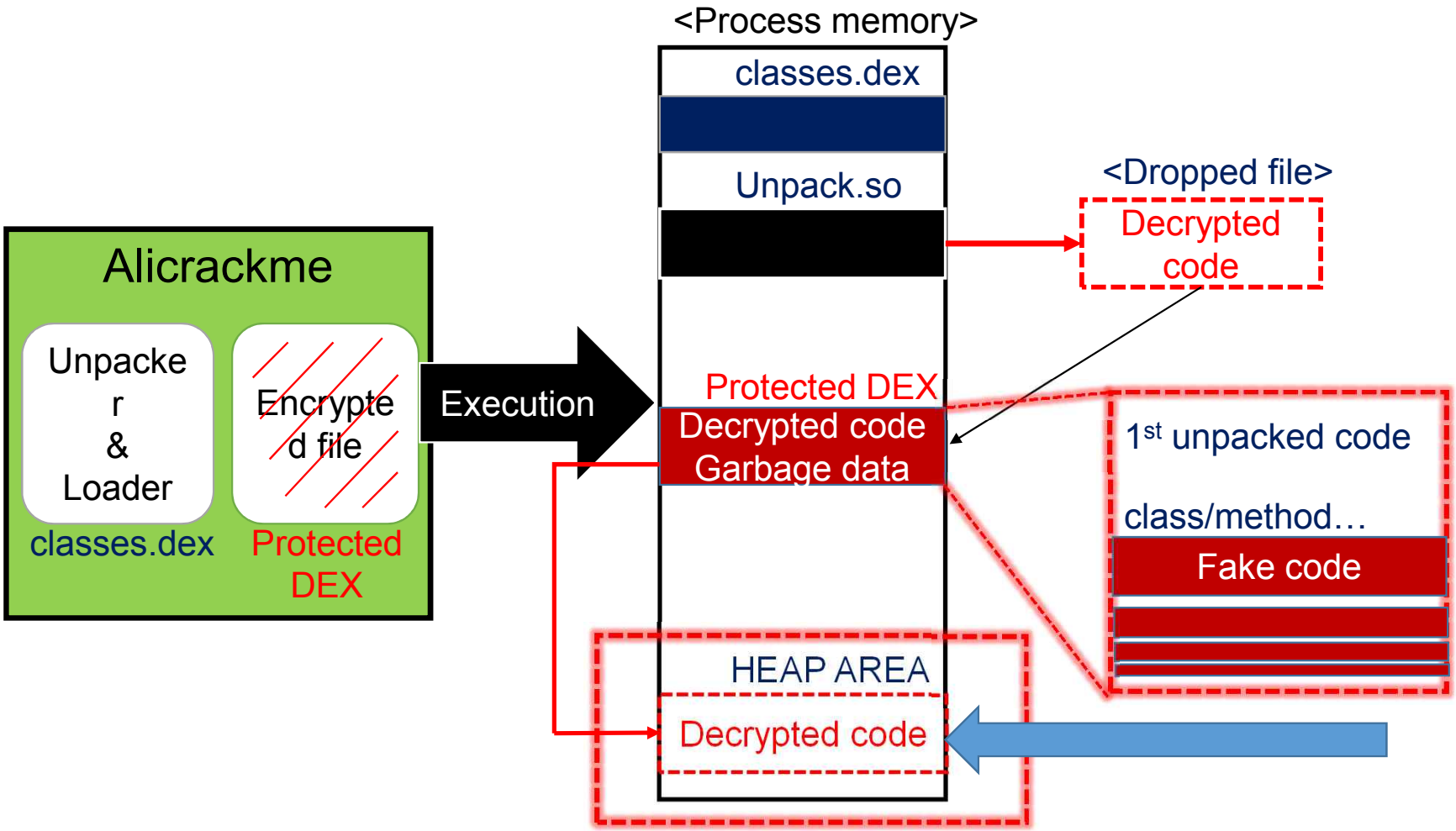



```
public class StubApplication extends Application {
    static {
        try {
            Class v2 = Class.forName("android.os.SystemProperties");
            Object v1 = v2.getDeclaredMethod("get", String.class).invoke(
        }
        catch(Exception v3) {
            v3.printStackTrace();
        }

        if(((String)v1).equalsIgnoreCase("x86")) {
            System.loadLibrary("mobisecx");
        }
        else {
            System.loadLibrary("mobisec");
        }
    }

    public StubApplication() {
        super();
    }

    protected native void attachBaseContext(Context arg1) {
    }

    public native void onCreate() {
    }
```

Advanced Packer



Advanced Packer



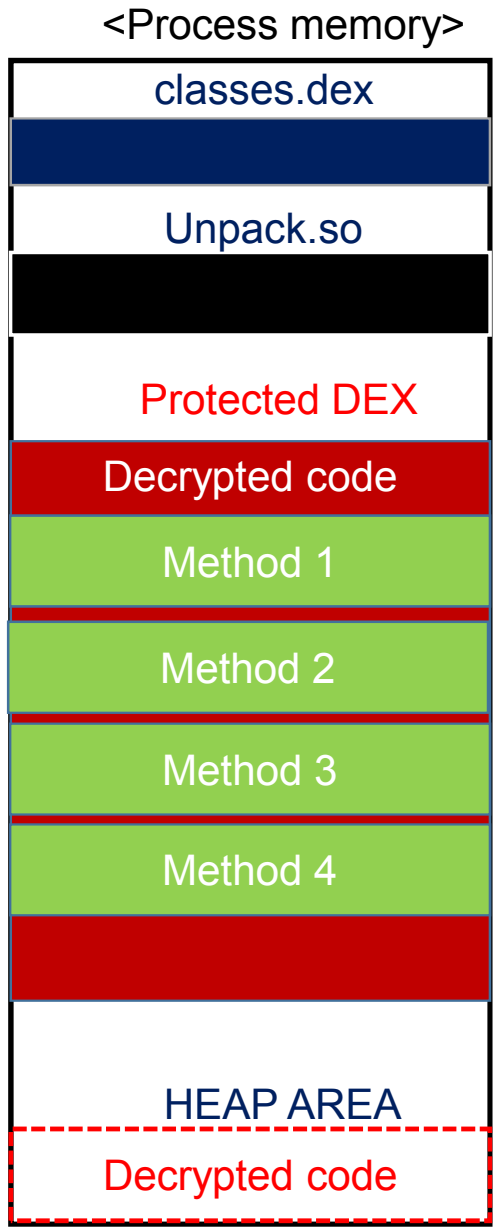
Advanced Packer

- Unpacking tools
  - DexHunter
  - Timing:

```
ClassObject* dvmDefineClass(Dv
```

```
struct DexOrJar {
    char*       fileName;
    bool        isDex;
    bool        okayToFree;
    RawDexFile* pRawDexFile;
    JarFile*    pJarFile;
    u1*         pDexMemory; //
};
```

```
/* shared memory region with file contents */
bool                isMappedReadOnly;
MemMapping          memMap;
```



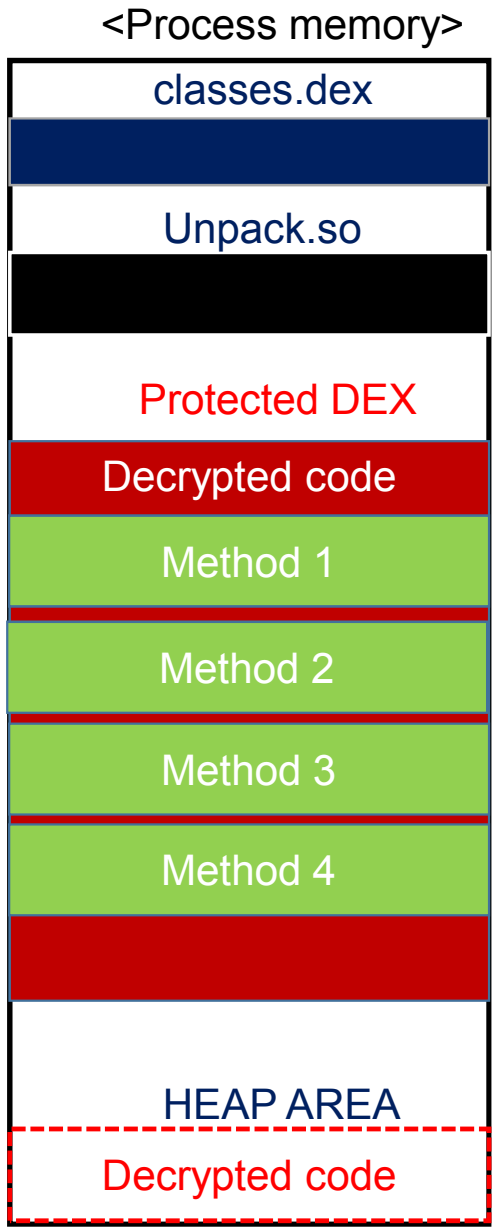
Advanced Packer

- Unpacking tools
  - AppSpear
  - Timing:

```
struct DexCode {
    u2  registersSize;
    u2  insSize;
    u2  outsSize;
    u2  triesSize;
    u4  debugInfoOff;
    u4  insnsSize;
    u2  insns[1];
```

```
static void loadMethodFromDex(ClassObject
    Method* meth)
```

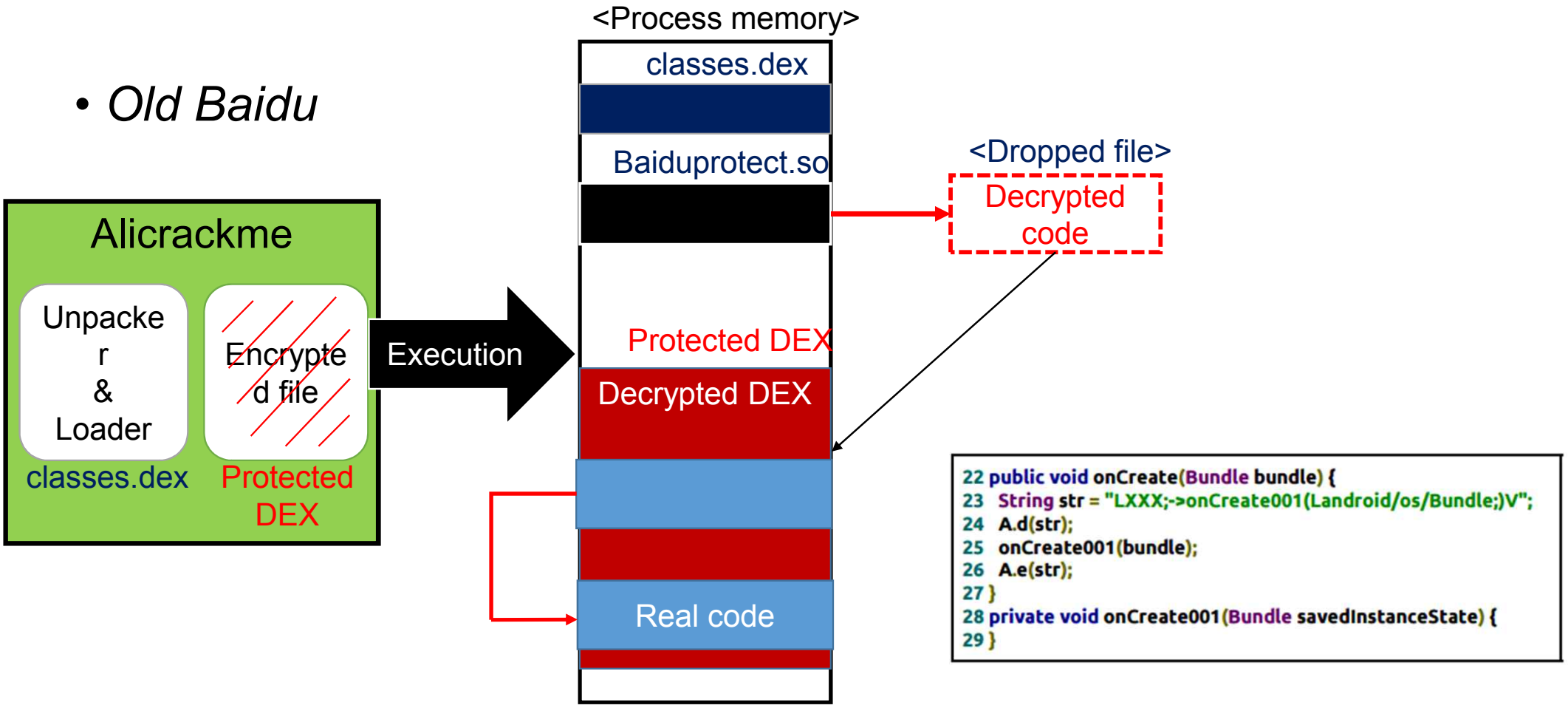
```
DEX_INLINE const DexCode* dexGetCode(const DexFile* pDexFile,
    const DexMethod* pDexMethod)
{
    if (pDexMethod->codeOff == 0)
        return NULL;
    return (const DexCode*) (pDexFile->baseAddr + pDexMethod->codeOff);
}
```



```
class b extends TimerTask {
    static {
        boolean v0 = !a.class.desiredAssertionStatus() ? true : false;
        b.a = v0;
    }

    b(a arg1, Handler arg2, String arg3) {
        // Decompilation failed
    }

    public void run() {
        String v0_2;
        dn.b(dn.a());
        if(Build$VERSION.SDK_INT < 10 || !Debug.isDebuggerConnected()) {
            String v5 = new e().a(this.c);
            if(v5.equals("sos")) {
                this.b.sendEmptyMessage(2);
            }
            else {
                CRC32 v0 = new CRC32();
                v0.update(v5.getBytes());
                v0.getValue();
                v5.hashCode();
                MessageDigest v0_1 = MessageDigest.getInstance("sha1");
                Cipher v1 = Cipher.getInstance("AES");
                if(!b.a && v1 == null) {
                    throw new AssertionError();
                }

                v1.init(2, new SecretKeySpec(Base64.decode("GXiQHT1CZ2elMzwpvvAoPA==".getBytes(), 0),
                        "AES"));
                new byte[0];
                byte[] v2 = v1.doFinal(Base64.decode("hjdsUjIT5je69WXIZP7Kzw==".getBytes("UTf-8"), 0));
                String v6 = new String(v2);
                v0_1.update(new byte[]{127});
                v0_1.update(v5.getBytes());
                v0_1.update(new byte[]{1});
```

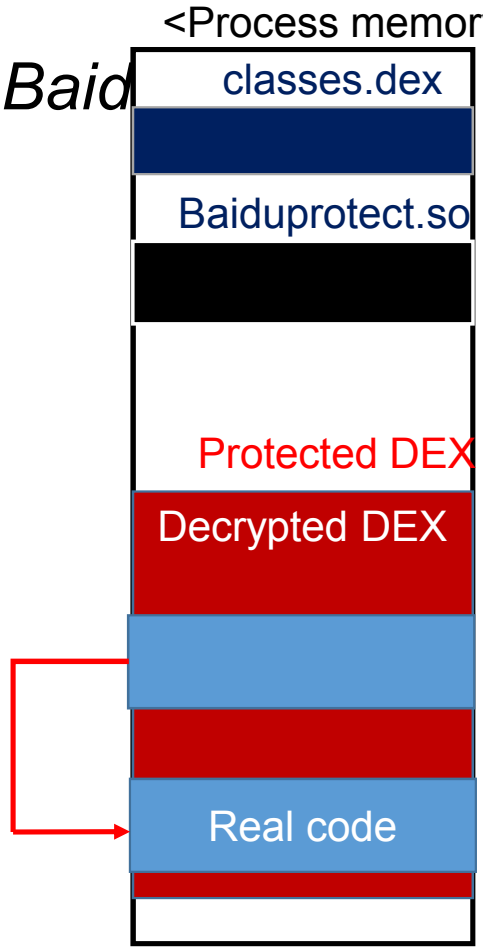
Advanced Packer

- *Old Baidu*

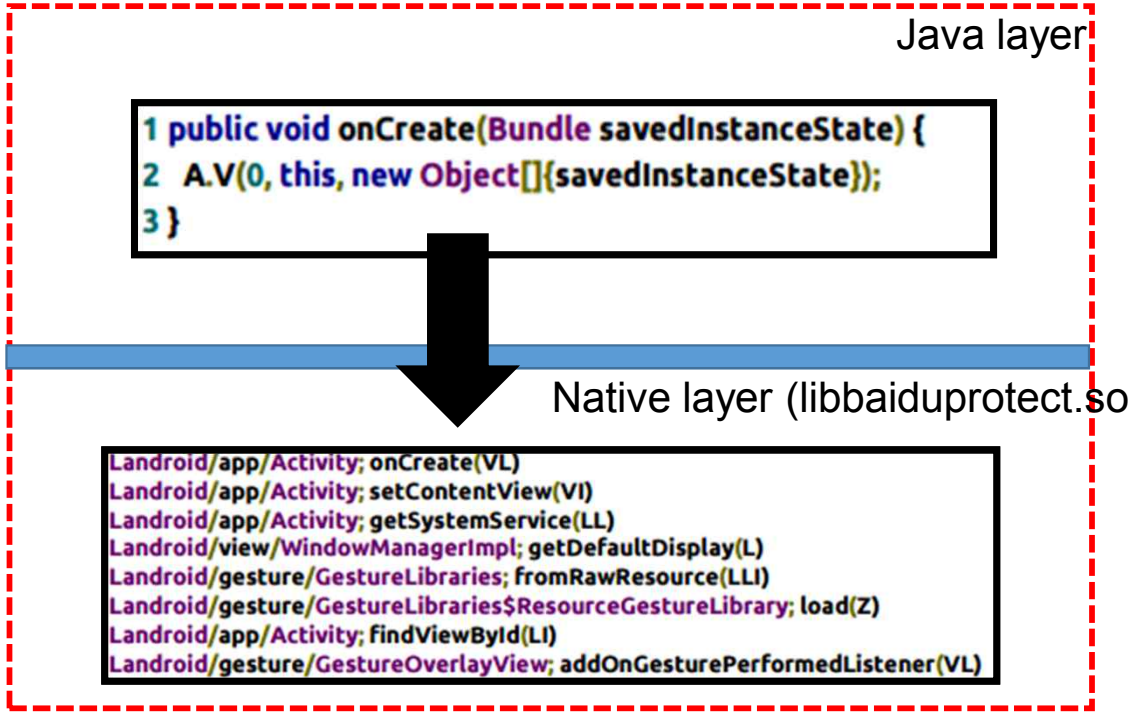


Advanced Packer

- *Recent Baid*




```
1 public void onCreate(Bundle savedInstanceState) {
2  super.onCreate(savedInstanceState);
3  setContentView(C0000R.layout.main);
4  this.display = ((WindowManager) getSystemService("window")).getDefaultDisplay();
5  this.mLibrary = GestureLibraries.fromRawResource(this, C0000R.raw.gestures);
6  if (!this.mLibrary.load()) {
7     finish();
8  }
9  findViewById(C0000R.id.gestures)).addOnGesturePerformedListener(this);
10 }
```



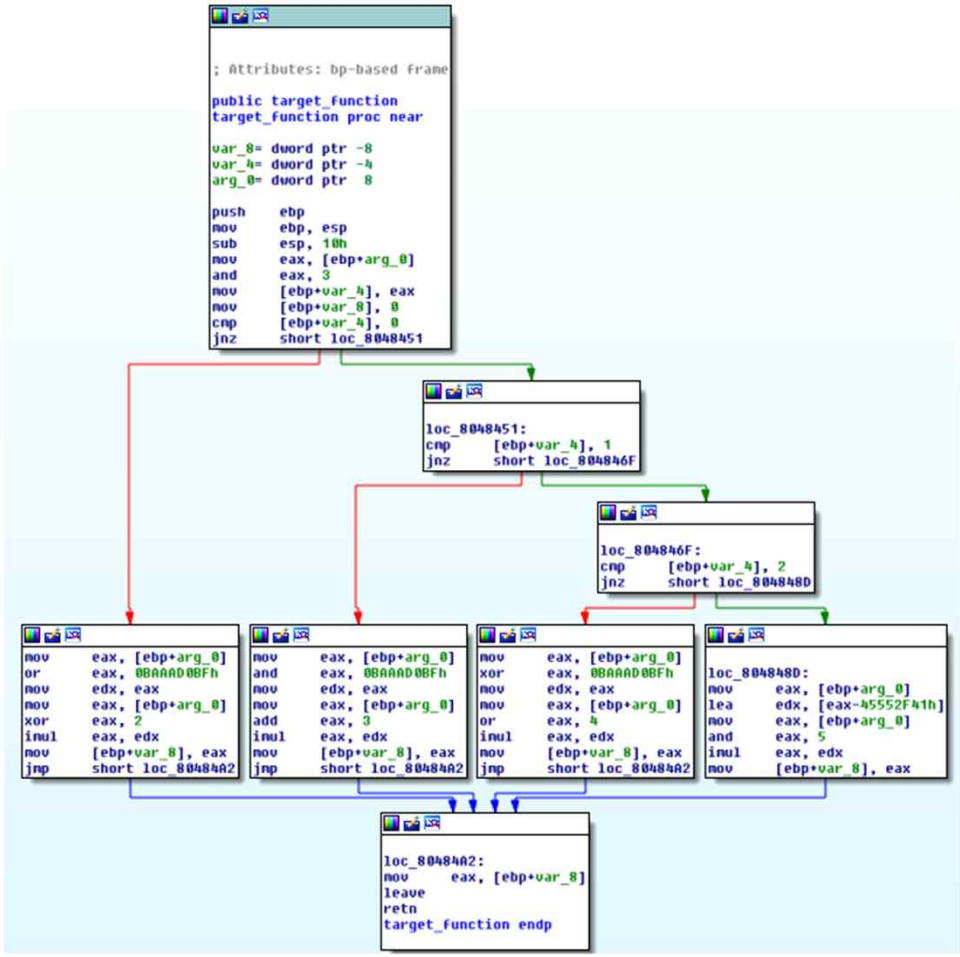
Obfuscation

- Open-source tools
- DexGuard
  - ClassEncryption
  - AssetEncryption
  - String Encryption
  - Symbol Name Obfuscation
  - Method call hiding
  - Native-code Encryption
  - Control-flow obfuscation

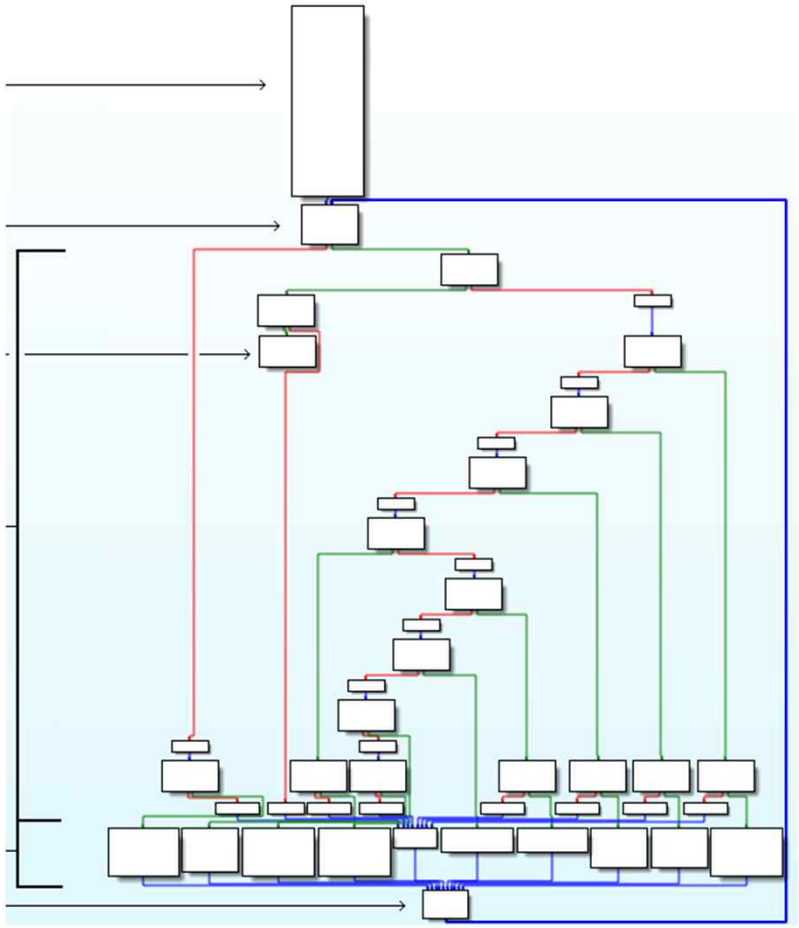
```
v1_2 = ((byte)v1_1[38]);
v3 = (v1_2 ^ -1) & 1 | v1_2 & -2;
v4 = (v1_2 & 1) << 1;
v2_1 = ((byte)((v3 ^ v4) + ((v3 & v4) << 1)));
v4 = (MainActivity.` ^ 21 | MainActivity.` & 21) << 1;
int v5 = MainActivity.` ^ 21;
MainActivity.= ((-v5 ^ v4) + ((v4 & -v5) << 1)) % 128;
try {
label_111:
    v3_1 = MainActivity.;
}
catch(ArrayStoreException v0) {
    goto label_147;
}

byte v3_2 = ((byte)v3_1[69]);
try {
    v1_3 = MainActivity.(v1_2, ((short)v2_1), v3_2);
    v1_3 = v1_3.intern();
}
```

```
v1 = new Object[3];
v1[2] = new Integer(v8);
v1[1] = this;
v1[0] = arg10;
Method v0_2 = Class.forName(HelloWorldActivity$if$.(-HelloWorldActivity$if$.-[60], 146,
        HelloWorldActivity$if$.-[18])).getDeclaredMethod(HelloWorldActivity$if$.(-HelloWorldActivity$if$
        .-[60], 186, HelloWorldActivity$if$.-[137]), String.class, ClassLoader.class, Integer
        .TYPE);
((AccessibleObject)v0_2).setAccessible(true);
return v0_2.invoke(null, v1);
```

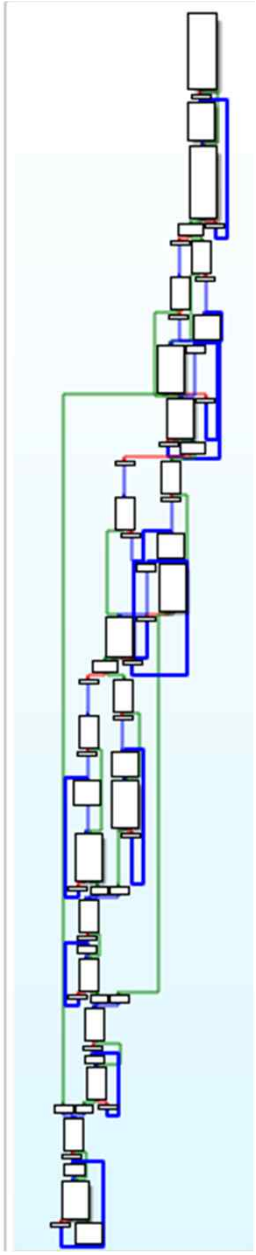
Obfuscation



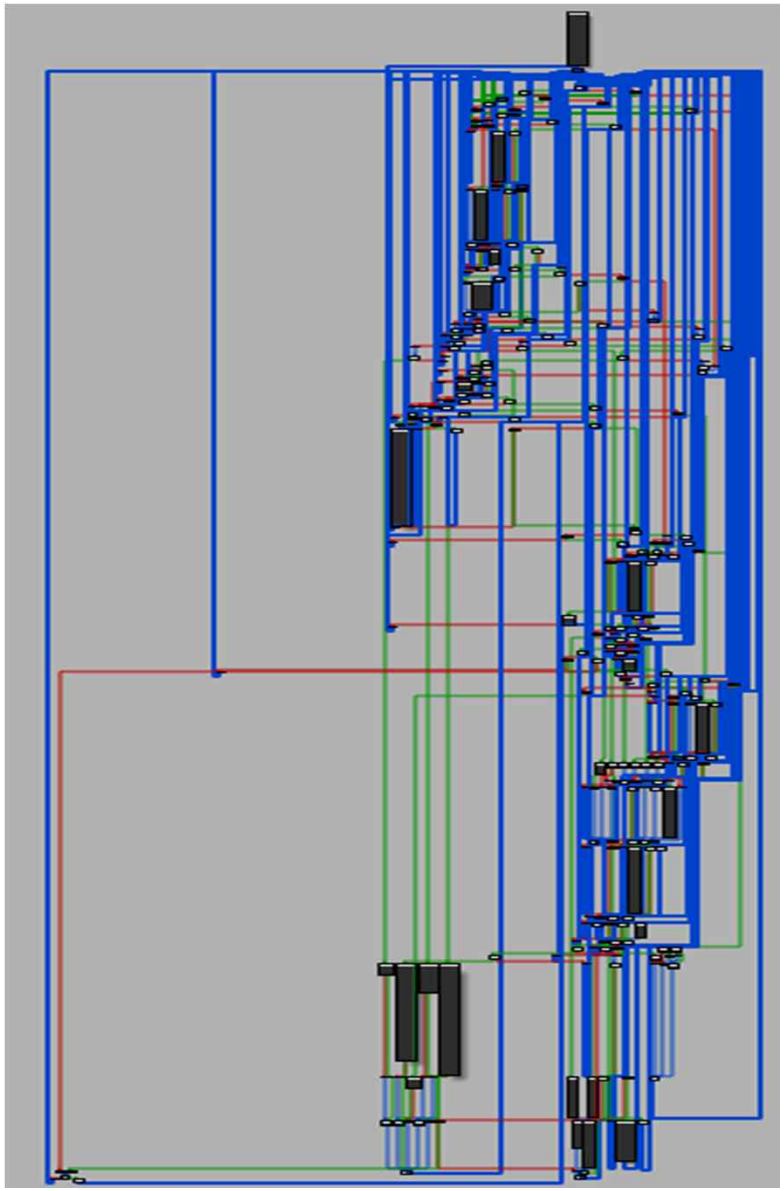
Obfuscation



Obfuscation

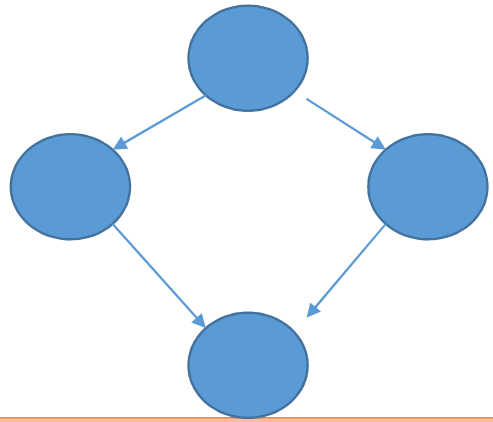


Obfuscation



```
}
if ( v2 > 2003265269 )
  break;
if ( v2 > -2085825462 )
{
  if ( v2 > -1932365830 )
  {
    if ( v2 > -1894470164 )
    {
      if ( v2 > -1876851337 )
      {
        if ( v2 > -1805753113 )
        {
          if ( v2 > -1779038877 )
          {
            if ( v2 > -1757798055 )
            {
              if ( v2 > -1698374356 )
              {
                if ( v2 > 1922088481 )
                {
                  if ( v2 == 1922088482 )
                  {
                    v12 = v70;
                    v67 = (((v12[11] << 8) | v12[10]) << 16) | (v12[9] << 8) | v12[8];
                    v76 = (int *)(3 * ((((v12[7] << 8) | v12[6]) << 16) | (v12[5] << 8) | v12[4]) * v67);
                    v13 = j_j_malloc((size_t)v76);
                    lsbbuff = (int)v13;
                    lsbbuffsize = (int)v76;
                    v13[3] = 0;
                    v13[2] = 0;
                    v13[1] = 0;
                    *v13 = 0;
                    goto LABEL_69;
                  }
                }
                else if ( v2 <= 1920673103 )
                {
                  if ( v2 > 1671106330 )
                  {
                    if ( v2 == 1671106331 )
                    {
                      v74 = (unsigned int)&v35;
                      v35 = 1634038374;
                      v76 = v1;
                      v36 = 100;
```

Obfuscation



```
public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Display the message.
        TextView view = new TextView(this);
        if(savedInstanceState != null) {
              view.setText("What?!");
        }
        else {
              view.setText("HUll");
        }

        view.setGravity(Gravity.CENTER);
        setContentView(view);

        // Briefly display a comment.
        Toast.makeText(this, "DexGuard has obfuscated the code of this sample", Toast.LENGTH_LONG).show();
    }
```

Obfuscation