ANDROID HACKER PROTECTION LEVEL 0

+ some blackphone stuff

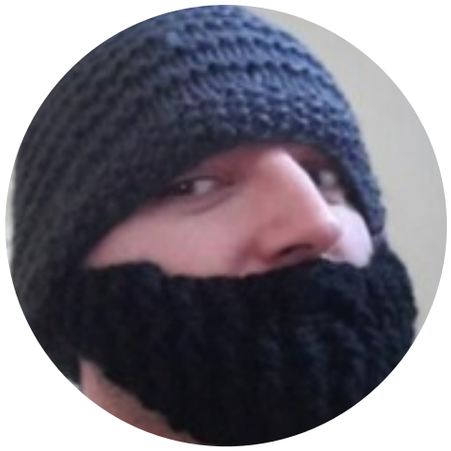
TIM “DIFF” STRAZZERE - JON “JUSTIN CASE” SAWYER

08.10.2014

Defcon 22

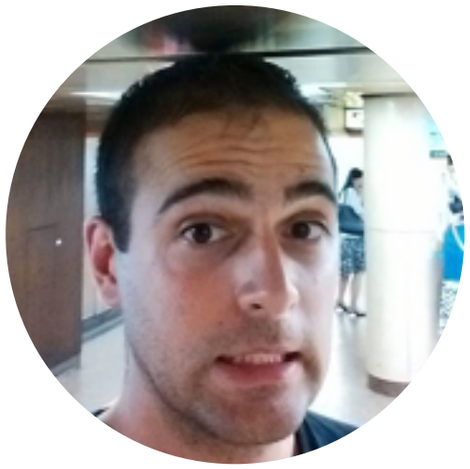
APPLIED CYBERSECURITY LLC

WHO ARE WE



JCASE

- CTO of Applied Cybersecurity LLC
- Professional Exploit Troll
- Has big mouth
- @TeamAndIRC
- github.com/CunningLogic



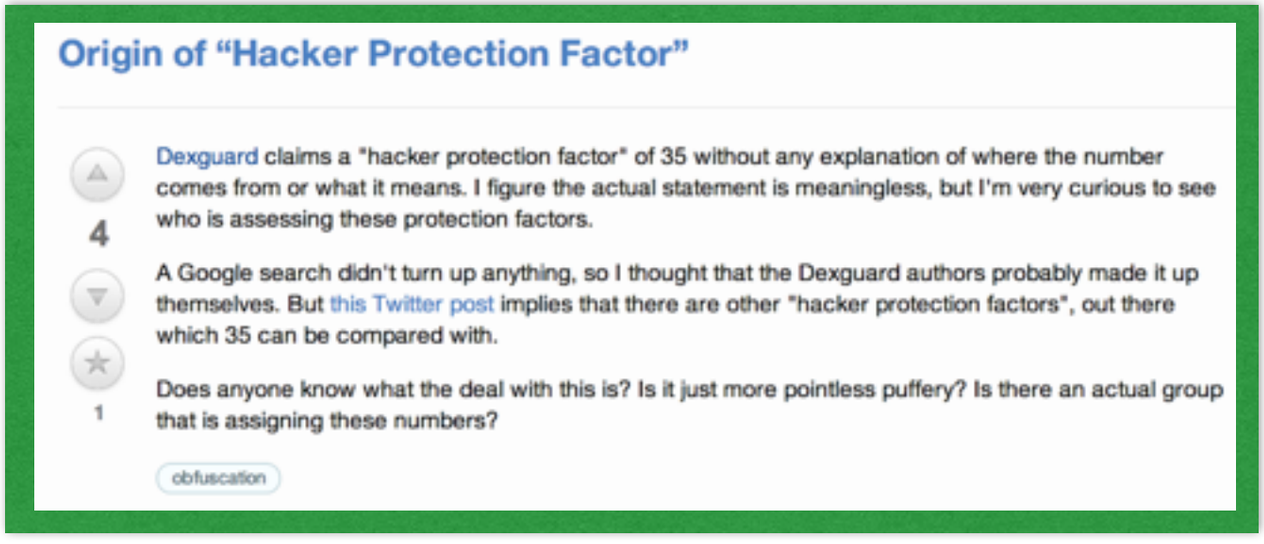
DIFF

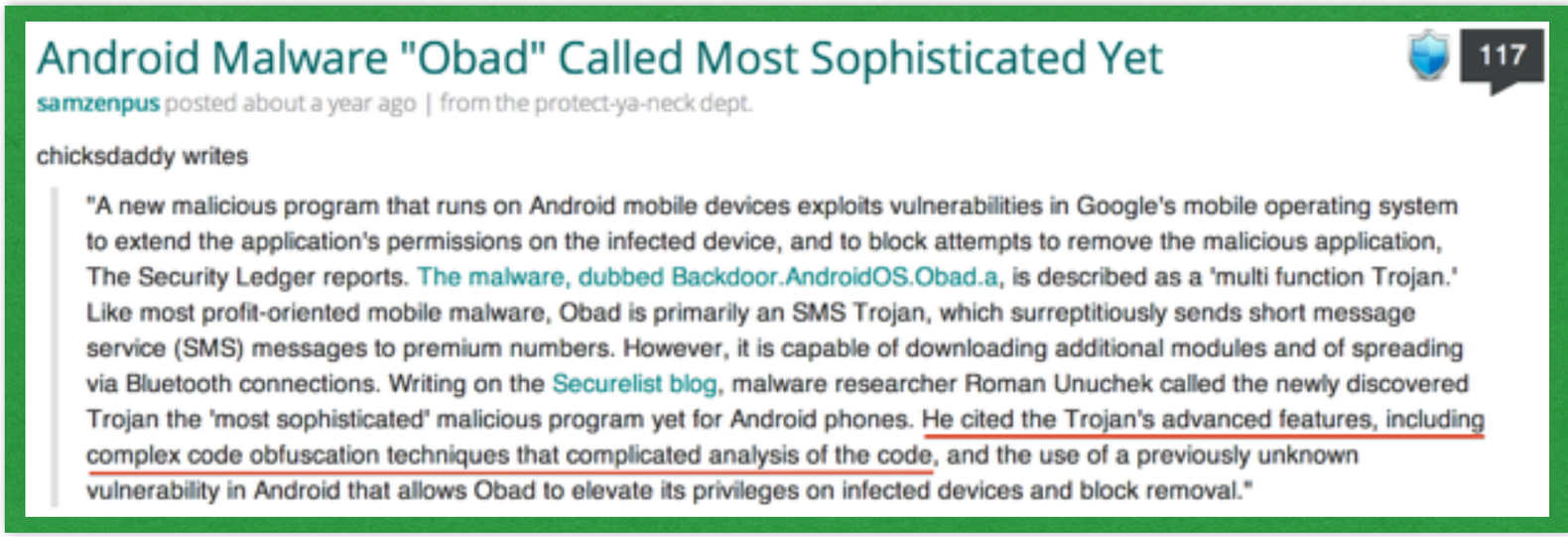
- Research & Response Engineer @ Lookout
- Obfuscation Junkie
- Pretends to know as much as JCase
- @timstrazz
- github.com/strazzere

WHY ARE WE HERE

More importantly - why should you care?

- Obfuscation is “magical”
- Quantifying the challenge is hard, mainly marketing material in Google results
- Good devs use it
- “Interesting” devs use it
- Bad devs use it
- Understanding apps is hard, let’s classify everything as bad and just blog!





“So good, even malware authors use us!”

WHAT IS OUT THERE

- Then -
  - Dex Education 101 - Blackhat 2012
    - Anti-decompilation tricks
    - Anti-analysis tricks
    - Demo/Release POC packer
  - General Optimizers / Minimal Obfuscators

- A little bit after…
  - Integration of tricks, release of specific tools
  - One off tools targeting environments/toolsets

- Now -
  - Most anti-decompilation/analysis tricks fixed in mainstream tools (baksmali, dex2jar, IDA Pro, radar)
  - Main stream commercial packers, protectors and obfuscates

PACKERS, PROTECTORS?

So - UPX and other stupid stuff?

- Optimizers / Obfuscators
  - Good practice for devs
  - Removes dead code / debug code
  - Potentially encrypt / obfuscate / hide via reflection

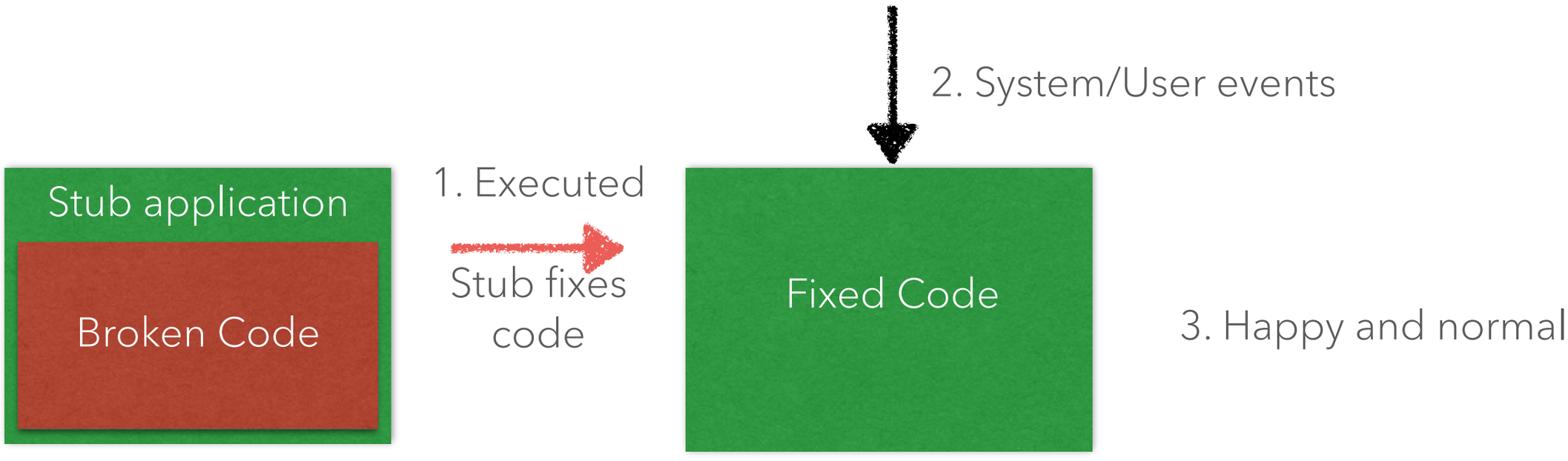
```
public void onClick(DialogInterface arg7, int arg8) {
    try {
        Class.forName("java.lang.System").getMethod("exit", Integer.TYPE).invoke(null, Integer.valueOf(0));
        return;
    } catch (Throwable throwable) {
        throw throwable.getCause();
    }
}
```

```
public void onClick(DialogInterface arg7, int arg8) {
    try {
        Class.forName(COn.´(-COn.`[0xC], COn.`[0x12], -COn.`[0x10])).getMethod(COn.´(i1, i2, i2 | 6), Integer.TYPE)
            .invoke(null, Integer.valueOf(0));
        return;
    } catch (Throwable throwable) {
        throw throwable.getCause();
    }
}
```

PACKERS, PROTECTORS?

So - UPX and other stupid stuff?

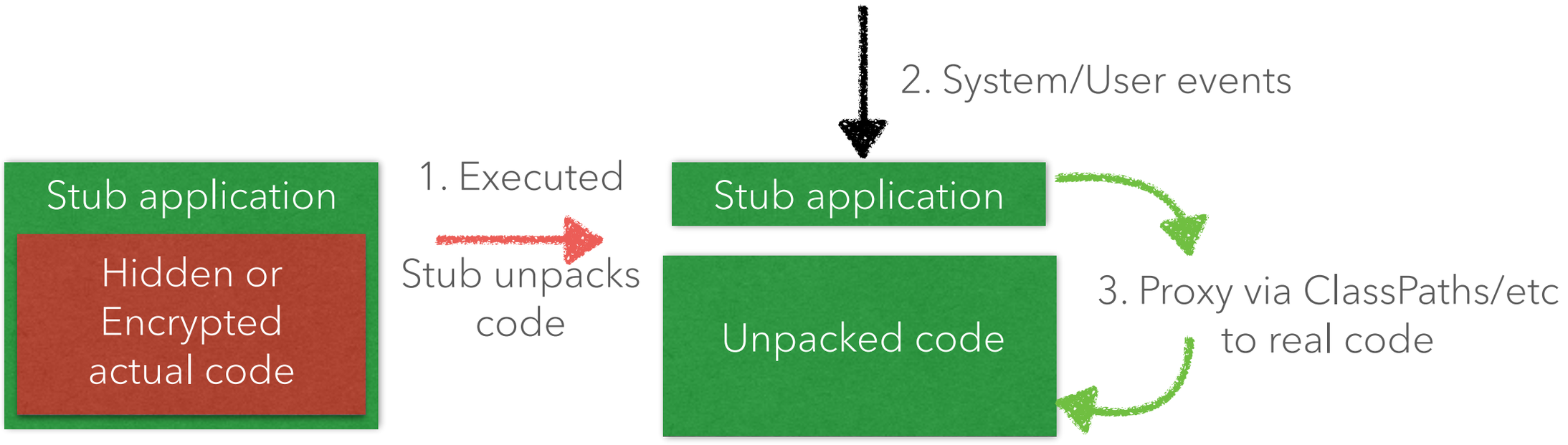
- "Protectors"
  - Classification similar to packers - manipulating "bad" code into workable things post execution
  - Performs anti-analysis/emulator tricks



PACKERS, PROTECTORS?

So - UPX and other stupid stuff?

- Packers
  - Similar to UPX and others - launcher stub and unfolding main application into memory
  - Performs anti-analysis/emulator tricks

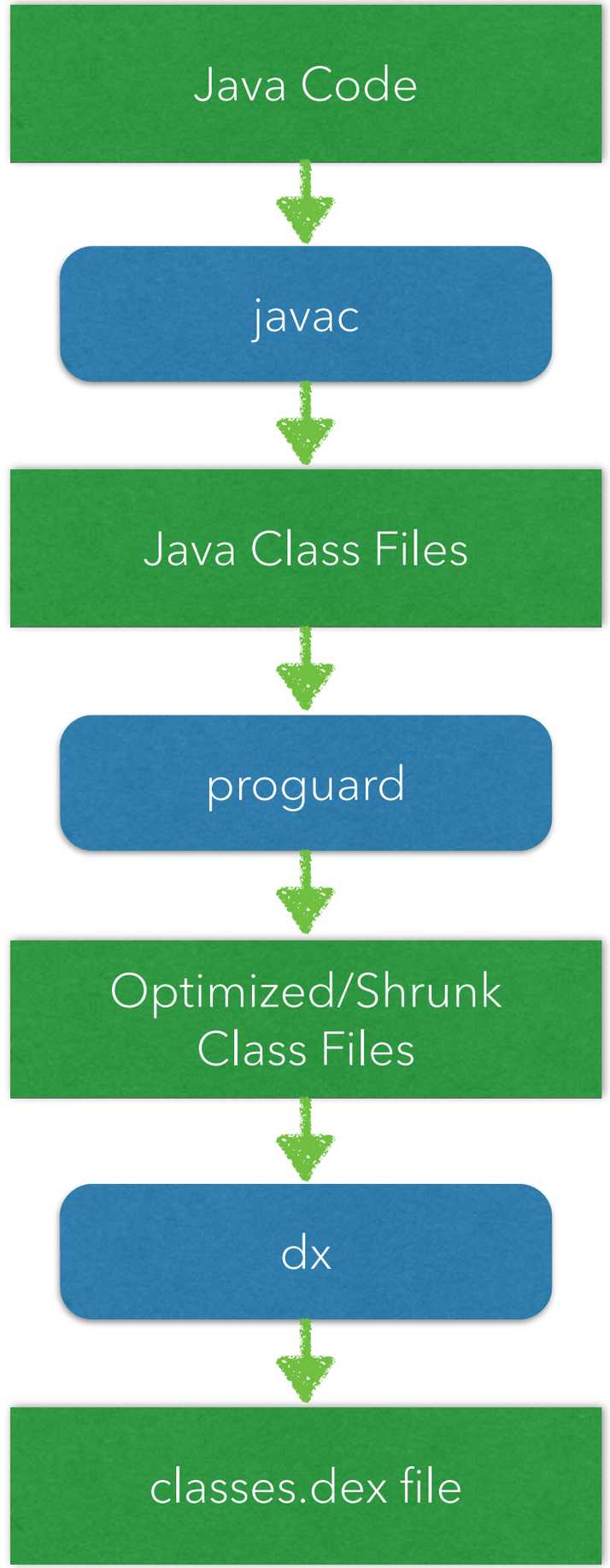


OPTIMIZERS & OBFUSCATORS

PROGUARD

Optimizers & Obfuscators

- ~8 years older than Android
- Created by Eric Lafortune
  - Specifically designed for Java
- Recommended By Google for Android developers
- Optimizer
- Shrinker
- Obfuscator (barely)
- Cost: $FREE
- Bundled in Android SDK



PROGUARD

Optimizers & Obfuscators

What does it do?

- Removes unnecessary/unused code
- Merges identical code blocks
- Performs ‘peep hole’ optimizations
- Removes debug information
- Renames objects (compacting names)
- Restructures code

PROGUARD

Optimizers & Obfuscators

Class Structure List

PROGUARD

Optimizers & Obfuscators

Class “source” Data
(debug info)

```
#------------------------------------------------------------------------------

.class public Utils
.super Object
.source "Utils.java"
```

```
#------------------------------------------------------------------------------

.class public ak
.super Object
.source “"
```

PROGUARD

Optimizers & Obfuscators

Line Numbers (debug info)

```
.method public static exec(String, Boolean)String
          .registers 12
          .param p0, "cmd"
          .param p1, "root"
          .prologue
          .line 163
          const-string            v7, "mksh"
          .line 164
          .local v7, shell:Ljava/lang/String;
          invoke-virtual          Boolean->booleanValue()Z, p1
          move-result             v8
          if-eqz                  v8, :14
:10
          .line 165
          const-string            v7, "su"
```

```
.method public static a(String, Boolean)String
          .registers 8
          const-string            v0, "mksh"
          invoke-virtual          Boolean->booleanValue()Z, p1
          move-result             v1
          if-eqz                  v1, :14
:10
          const-string            v0, "su"
```

PROGUARD

Optimizers & Obfuscators

Original Java Source

```
public static String exec(String cmd, Boolean root) {
     BufferedReader bufferedReader;
     DataOutputStream dataOutputStream;
     Process process;
     String string = "sh";
     if(root.booleanValue()) {
         string = "su";
     }

     StringBuilder stringBuilder = new StringBuilder();
     try {
         process = Runtime.getRuntime().exec(string);
         dataOutputStream = new DataOutputStream(process.getOutputStream());
         dataOutputStream.writeBytes(cmd + "\n");
         bufferedReader = new BufferedReader(
                 new InputStreamReader(process.getInputStream()));
     }
     catch(IOException iOException) {
         goto label_36;
     }

     try {
         dataOutputStream.writeBytes("exit\n");
         dataOutputStream.flush();
         String string1 = System.getProperty("line.separator");
         while(true) {
             String string2 = bufferedReader.readLine();
             if(string2 == null) {
                 break;
             }

             stringBuilder.append(string2);
             stringBuilder.append(string1);
         }
```

Decompiled ProGuarded Output

```
public static String a(String arg6, Boolean arg7) {
    Process process;
    String string = "mksh";
    if(arg7.booleanValue()) {
        string = "su";
    }

    StringBuilder stringBuilder = new StringBuilder();
    try {
        process = Runtime.getRuntime().exec(string);
        DataOutputStream dataOutputStream = new DataOutputStream(
                process.getOutputStream());
        dataOutputStream.writeBytes(String.valueOf(arg6) + "\n");
        BufferedReader bufferedReader = new BufferedReader(
                new InputStreamReader(process.getInputStream()));
        dataOutputStream.writeBytes("exit\n");
        dataOutputStream.flush();
        String string1 = System.getProperty("line.separator");
        while(true) {
            String string2 = bufferedReader.readLine();
            if(string2 == null) {
                break;
            }

            stringBuilder.append(string2);
            stringBuilder.append(string1);
        }
```

PROGUARD

Optimizers & Obfuscators

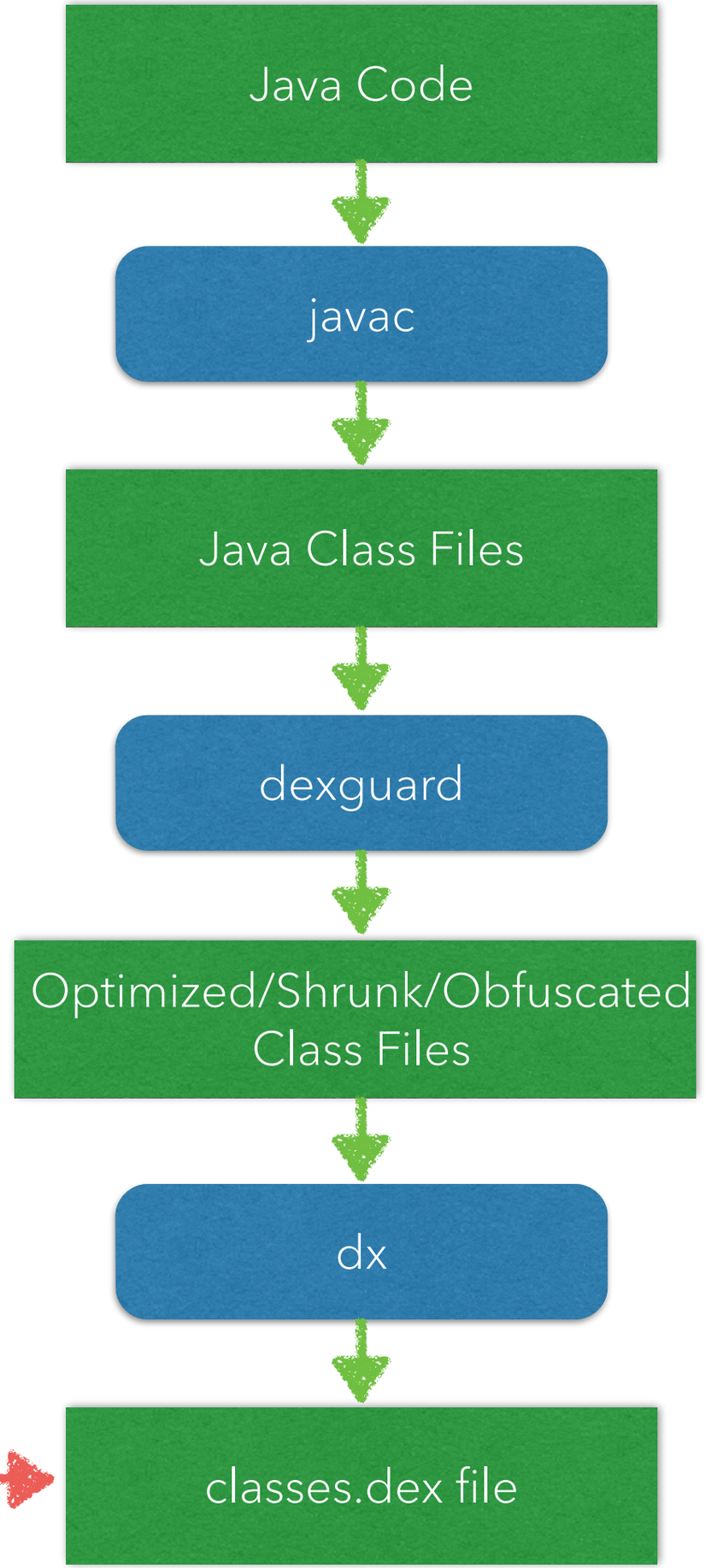
What is it good for?

- Decreases dex file size
- Increases app speed/performance
- Decreases memory usage
- Removes debug information
  (slightly increase reversing complexity)
- Doesn’t do much obfuscation
- “Hacker Protection Factor 0”

DEXGUARD

Optimizers & Obfuscators

- Son of ProGuard
- Create by Eric Lafortune
- “Standard” protection
- Optimizer
- Shrinker
- Obfuscator/Encryptor
- Cost: $650 - $1300



DEXGUARD

Optimizers & Obfuscators

What does it do?

- Everything ProGuard does
- Automatic reflection
- String encryption
- Asset & library encryption
- Class encryption (packing)
- Application tamper detection

DEXGUARD

Optimizers & Obfuscators

```
public void onClick(DialogInterface arg2, int arg3) {
     System.exit(0);
}
```

Automatic Reflection

```
public void onClick(DialogInterface arg7, int arg8) {
     try {
          Class.forName("java.lang.System").getMethod("exit", Integer.TYPE).invoke(null, Integer.valueOf(0));
          return;
     } catch (Throwable throwable) {
          throw throwable.getCause();
     }
}
```

String Encryption

```
public void onClick(DialogInterface arg7, int arg8) {
     try {
          Class.forName(COn.´(-COn.`[0xC], COn.`[0x12], -COn.`[0x10])).getMethod(COn.´(i1, i2, i2 | 6), Integer.TYPE)
               .invoke(null, Integer.valueOf(0));
          return;
     } catch (Throwable throwable) {
          throw throwable.getCause();
     }
}
```

DEXGUARD

Optimizers & Obfuscators

String Encryption

Original Code

```
public static void kaBoom(Context context) {
     while(true) {
          context.sendStickyBroadcast(new Intent("android.net.wifi.STATE_CHANGE"));
     }
}
```

Encrypted Strings in Main Array

```
MainActivity.鷂 = new byte[]{0x4C, 0xE, 2, 9, -7, 0x10, -54, 0x3E, 0x17, -9, -44, 0x4C, 0xA, 0x1B, -7, 0x13, -98,
     6, 0x1B, -76, 0x17, 0x4C, 0xE, 2, 9, -7, 0x10, -54, 0x3E, 0x17, -9, -44, 0x42, 0xD, 0xD, 9, -11, 0x13, 8,
     -55, 0x4D, -5, 6, 0x14, 0xF, -9, 0x15, 0xF, -67, 0x4D, -3, -64, 0x26, -22, 0x17, 0x4C, 0xE, 2, 9, -7, 0x10,
     -54, 0x3E, 0x17, -9, -44, 0x42, 0xD, 0xD, 9, -11, 0x13, 8, -55, 0x4D, -5, 6, 0x14, 0xF, -9, 0x15, 0xF, -67,
     0x4D, -3, -50, 0x3C, 7, 0xA, 0x10, 0x12, 3, 0xA, 3, -3, 0x15, 9, 0xA, -63, 0x3B, 0x17, 3, 5, 8, 0xD, -62,
     0x4D, -3, 0x12, 0x19, 4, 3, 0xD, 1, 0x2E, 0x13, -5, 8, -3, 0xD, 0xD, 0xA, -3, -60, 0x1F, 6, 8, -13, 0x17,
     0x4C, 0xE, 2, 9, -7, 0x10, -54, 0x3E, 0x17, -9, -44, 0x42, 0xD, 0xD, 9, -11, 0x13, 8, -55, 0x4D, -5, 6,
     0x14, 0xF, -9, 0x15, 0xF, -67, 0x4D, -3, 3, -69, 0x17, 0x3D, 5, 0x1B, -11, -42, 0x45, -3, 0x1A, 9, -13 … };
```

New Obfuscated Code

```
public static void 鷂(MainActivity arg4) {
     while(true) {
          ((Context)arg4).sendStickyBroadcast(new Intent(MainActivity.鷂(0xFFFFFE84, 0x23, 0x276)));
     }
}
```

DEXGUARD

Optimizers & Obfuscators

String Encryption Code Example

Obfuscated Decryption Function

```
private static String 鷂(int arg6, int arg7, int arg8) {
     int i2;
     int i1;
     arg7 += 0x3E;
     byte[] array_b = MainActivity.鷂;
     int i = 0;
     arg6 += 0x199;
     byte[] array_b1 = new byte[arg6];
     if(array_b == null) {
     i1 = arg6;
     i2 = arg8;
     } else {
          label_12:
          array_b1[i] = ((byte)arg7);
          ++i;
          if(i >= arg6) {
               return new String(array_b1, 0);
          } else {
               i1 = arg7;
               i2 = array_b[arg8];
          }
     }

     ++arg8;
     arg7 = i1 + i2 - 8;
     goto label_12;
}
```

Deobfuscated

```
private static String decrypt(int length,
                              int cChar,
                              int pos) {
     int i = 0;
     int j = 0;
     int k = 0;
     cChar += 0x3E;
     length += 0x199;
     byte[] arrENC = new byte[length];
     while(i < length) {
          arrENC[i] = ((byte)cChar);
          k = cChar;
          if(pos < STRINGS.length)
               j = STRINGS[pos];
          ++pos;
          cChar = k + j - 8;
          ++i;
     }
     return new String(arrENC, 0);
}
```

DEXGUARD

Optimizers & Obfuscators

Asset & Library Encryption

```
AssetManager assetManager = context.getAssets();
File output = new File("/data/data/com.cunninglogic.bookexample/temproot");
InputStream inputStream = assetManager.open("temproot");
Cipher cipher = Cipher.getInstance(“AES/CFB/NoPadding");

byte[] myKey = new byte[]{-114, -53, -9, -86, -13, -14, -115, 0x6F, -41, -39,
     5, 0x28, -46, 0x74, -10, -20};
SecretKeySpec secretKeySpec = new SecretKeySpec(myKey, “AES");

// Initialization vector
byte[] myIV = new byte[]{-69, 0x49, -91, -7, -53, 2, -71, -97, -48, 0x62, -71,
     0x78, 0x11, -90, -85, -107};
int i = myIV[7] & 0x2D;
myIV[i] = ((byte)(i | 0x52));

cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, myIV);
CipherInputStream cipherInputStream = new CipherInputStream(inputStream,
     cipher);
FileOutputStream fileOutputStream = new FileOutputStream(output);
byte[] buf = new byte[1024];
int read;
while(read = cipherInputStream.read(buf) |= -1) {
     fileOutputStream.write(buf, 0, read);
}

inputStream.close();
cipherInputStream.close();
fileOutputStream.flush();
fileOutputStream.close();
```

DEXGUARD

Optimizers & Obfuscators

Class Encryption

```
File output = new File(“/out/put/path/decrypted.zip"); // Path to write zipfile to

byte[] myKey = new byte[]{ ... }; // Key
byte[] myIV = new byte[]{ ... }; // IV
byte[] encDex = new byte[]{ ... }; // Encrypted zip/dex

int inputLen = 0x7FD; // inputLen
int inputOffset = 0x14; // inputOffset

Cipher cipher = Cipher.getInstance("AES/CFB/NoPadding");

SecretKeySpec secretKeySpec = new SecretKeySpec(myKey, "AES");
IvParameterSpec ivSpec = new IvParameterSpec(myIV);
cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivSpec);
byte[] decDex = cipher.doFinal(encDex, inputOffset, inputLen);
```

DEXGUARD

Optimizers & Obfuscators

Class Encryption

```
byte[] zipHeader = new byte[]{0x50, 0x4B, 0x03, 0x04};
byte[] zipbuf = new byte[4];
int i = 0;
for (i = 0; i < decDex.length - 3; ++i) { // Locate header of the zip file
     zipbuf[0] = decDex[i];
     zipbuf[1] = decDex[i + 1];
     zipbuf[2] = decDex[i + 2];
     zipbuf[3] = decDex[i + 3];
     if (Arrays.equals(zipHeader, zipbuf)) {
          break;
     }
}

byte[] outDex = new byte[decDex.length - i];
int j = 0;
while (!(j == outDex.length)) {
     outDex[j] = decDex[i];
     ++j;
     ++i;
}

ByteArrayInputStream bis = new ByteArrayInputStream(outDex);
FileOutputStream fileOutputStream = new FileOutputStream(output);
byte[] buf = new byte[4*1024];
int read;
while ((read = bis.read(buf)) != -1) {
     fileOutputStream.write(buf, 0, read);
}
```

DEXGUARD

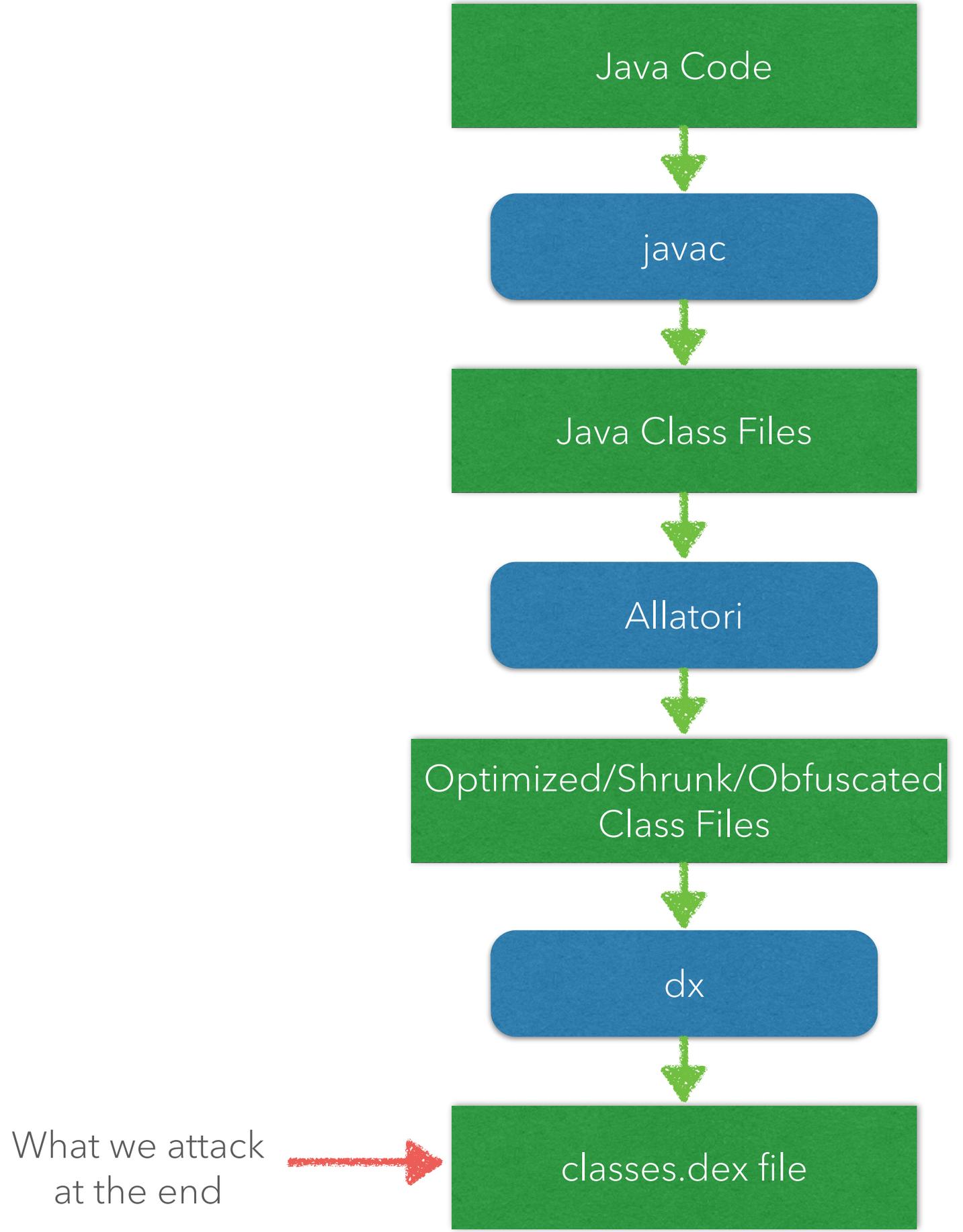
Optimizers & Obfuscators

- May increase dex file size
- May decrease app speed
- May increase memory usage
- Removes debug information
- Automatic string encryption
- Asset, Library, Class encryption
- Best Feature: Automatic reflection (combined with string enc)
- Moderately priced & easy to use
- Reversible with moderate effort
- “Hacker Protection Factor 1”

ALLATORI

Optimizers & Obfuscators

- Optimizer
- Shrinker
- Obfuscator
- Watermarker
- Cost: $290
- Free Academic Version



ALLATORI

Optimizers & Obfuscators

What does it do?

- Name obfuscation
- Control flow flattening/obfuscation
- Debug info obfuscation
- String encryption

ALLATORI

Optimizers & Obfuscators

```
public class OnBootReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent){
    if (!new File("/system/xbin/su").exists()) {
        if (new File("/data/data/com.cunninglogic.weaksauce/temp/onboot").exists())
            Weak.peppers(context);
    }
    }
}
```

```
public void onReceive(Context arg0, Intent arg1) {
    if(!new File(K.A("L\u0019\u001A\u0019\u0017\u000F\u000EE\u001B\b\n\u0004L\u0019\u0016")).exists() &&
        (new File(K.A("L\u000E\u0002\u001E\u0002E\u0007\u000B\u0017\u000BL\t\f\u0007M\t\u0016\u0004\r\u0003\r\r
\u000F\u0005\u0004\u0003\u0000D\u0014\u000F\u0002\u0001\u0010\u000B\u0016\t\u0006E\u0017\u000F\u000E\u001AL
\u0005\r\b\f\u0005\u0017")).exists())) {
            Weak.L(arg0);
    }
}
```

ALLATORI

Optimizers & Obfuscators

Obfuscated Encryption Function

```java
public static String A(String arg0) {
     int i = arg0.length();
     char[] array_ch = new char[i];
     --i;
     int i1;
     for(i1 = i; i >= 0; i1 = i) {
          int i2 = i1 - 1;
          array_ch[i1] = ((char)(arg0.charAt(i1) ^ 0x63));
          if(i2 < 0) {
               break;
          }

          i = i2 - 1;
          array_ch[i2] = ((char)(arg0.charAt(i2) ^ 0x6A));
     }

     return new String(array_ch);
}
```

Deobfuscated Encryption Function

```java
public static String decrypt(String enc_text) {
     int length = enc_text.length();
     char[] plaintext = new char[length];
     --length;
     int i;
     for(i = length; length >= 0; i = length) {
          int j = i - 1;
          plaintext[i] = ((char)(enc_text.charAt(i) ^ 0x63));
          if(j < 0)
               break;
          }

          length = j - 1;
          plaintext[j] = ((char)(enc_text.charAt(j) ^ 0x6A));
     }

     return new String(plaintext);
}
```

ALLATORI

Optimizers & Obfuscators

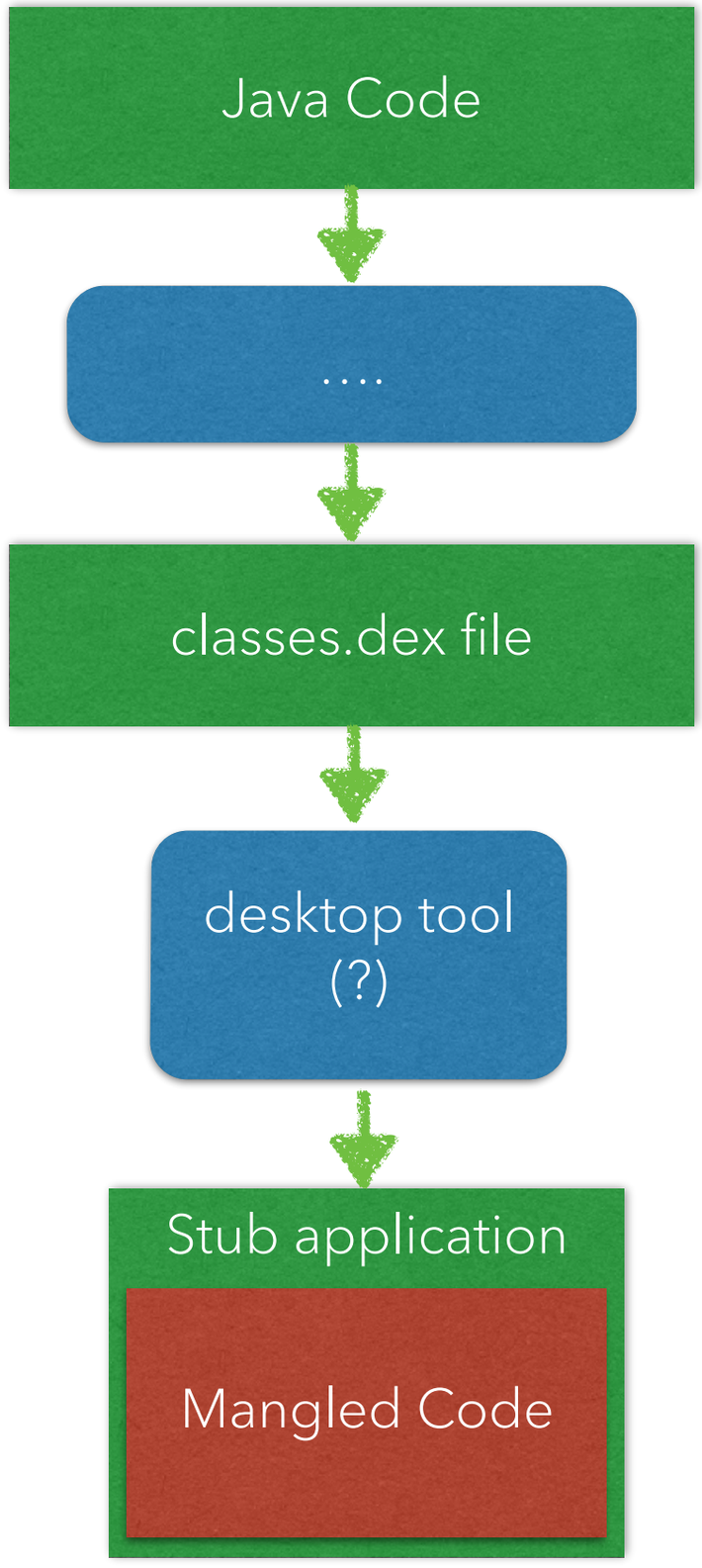
- Free licenses for educational use!
- Decreases dex file size
- Increases app speed
- Decreases memory usage
- Removes debug code
- Doesn’t do much in the ways of obfuscation
- “ProGuard + string encryption”
- Easily reversed
- “Hacker Protection Factor 0.5”

"PROTECTORS"

APKPROTECT

Protectors

- Chinese Protector
- Multiple iterations and rebrandings
    - DexCrypt / APKProtect (Lite, PC, Advanced)
- “Appears” active
- Anti-debug
- Anti-decompile
- Almost like a packer
- String encryption
- Cost: $Free - $Expensive (Site non-functional)



APKPROTECT

Protectors

- Tool mangles original code
  - Modifies entry point to loader stub
  - Prevents static analysis
- During runtime loader stub is executed
  - Performs anti-emulation
  - Performs anti-debugging
  - Fixes broken code in memory

Injected entrypoint inside chargeware/malware sample

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:versionCode="1"
     android:versionName="1.0"
     package="tyuyu.truртyr.rgreuyt4"
     >
     <uses-permission
     android:name="android.permission.SEND_SMS">
…
     <application
          android:theme="@7F070001"
          android:label="@7F060000"
          android:icon="@7F020000"
          android:name="APKPMainAPP11177"
          android:allowBackup="true"
          >
```

Mangled code as seen during static analysis

```
# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .registers 10

    :cond_ffffffffffff877e
    :cond_ffffffffffffd82f
    :cond_ffffffffffffff4fd
    return-void

    #Field index out of bounds: 55307
    #iget-short v3, p7, field@55307
    nop

    #Field index out of bounds: 17222
    #iput-char v0, p7, field@17222
    nop

    #Field index out of bounds: 19189
    #iget-boolean v5, v4, field@19189
    nop

    shr-int p81, p156, p183
```

Dalvik stub code, calling native stub

```
.class public Ltyuyu/truртyr/rgreuyt4/APKPMainAPP11177;
.super Landroid/app/Application;


# direct methods
.method public constructor <init>()V
    .registers 2

    invoke-direct {p0}, Landroid/app/Application;-><init>()V

    const-string v0, "APKProtect"

    invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V

    return-void
.end method
```

APKPROTECT

Protectors

1. Dalvik Optimizes the Dex file into memory, ignoring “bad” parts
2. Upon execution Dalvik code initiates, calls the native code
3. Native code fixes Odex in memory
4. Execution continues as normal

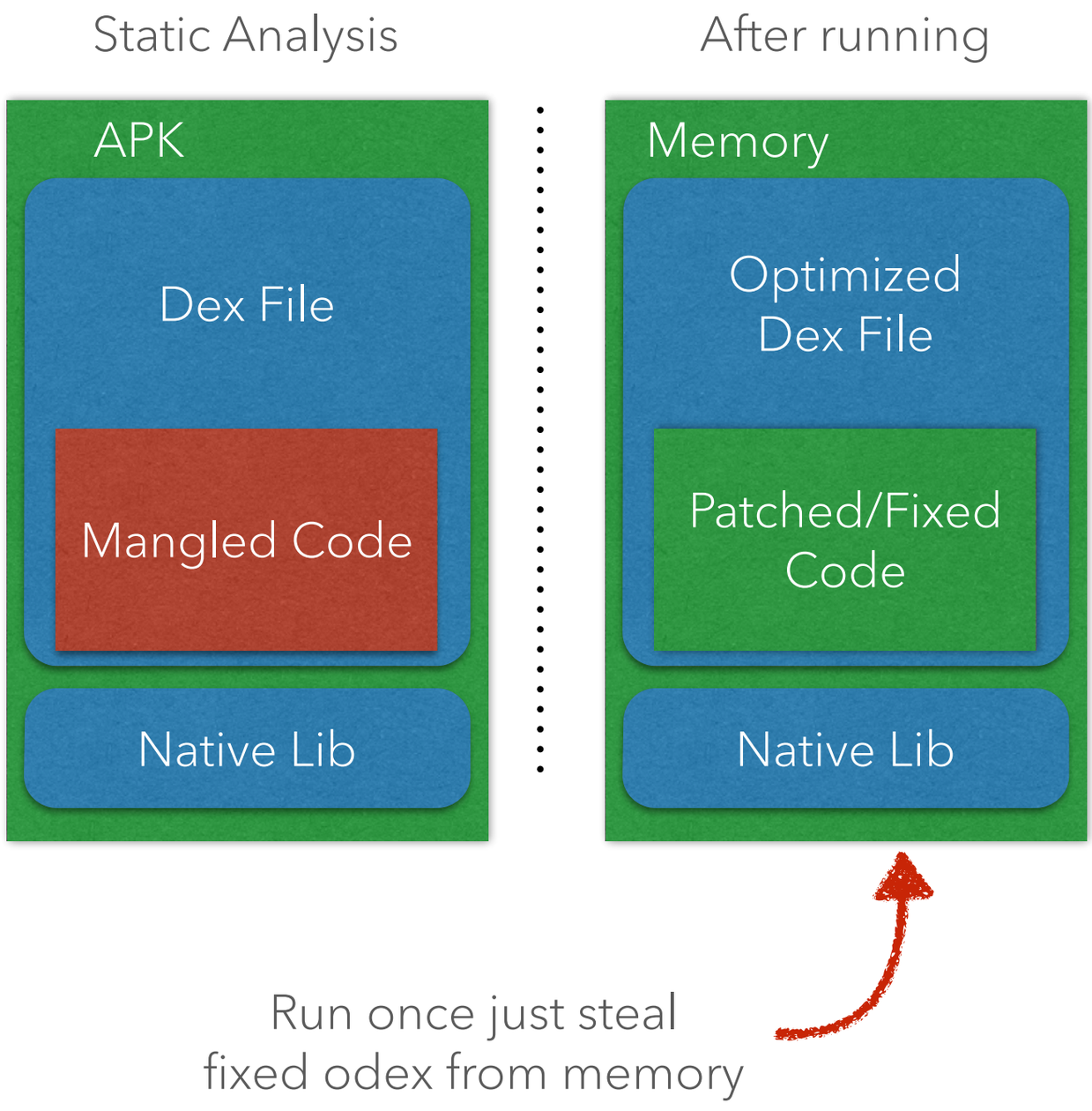
```
JNI_onLoad {
    ptrace(PTRACE_TRACEME, 0, NULL, NULL) //anti-debug
    if(!find_odex_file()) // anti-analysis
        create_infinitely_sleeping_thread();
    if(find_qemud_process()) // anti-emulation
        create_infinitely_sleeping_thread();
    patch_odex();
    return JNI_VERSION_1_6;
}
```

```
find_qemud_process() {
    for(int i = 0; i < 0x65; i++)
        if( hash(read(“/proc/%d/cmdline”, i))
            == hash(“/system/bin/qemud”))
        return true;
    return false;
}
```

APKPROTECT

Protectors

- Winning is easy!
- Avoid using QEMU or use LD_PRELOAD hack released with talk (nerf strlen() when assessing /system/bin/qemud)
- Attach to cloned process (no ptrace worries)
- Dump odex, de-odex with baksmali
- Reverse modified Base64 + DES string encryption
- Have the original code!



APKPROTECT

Protectors

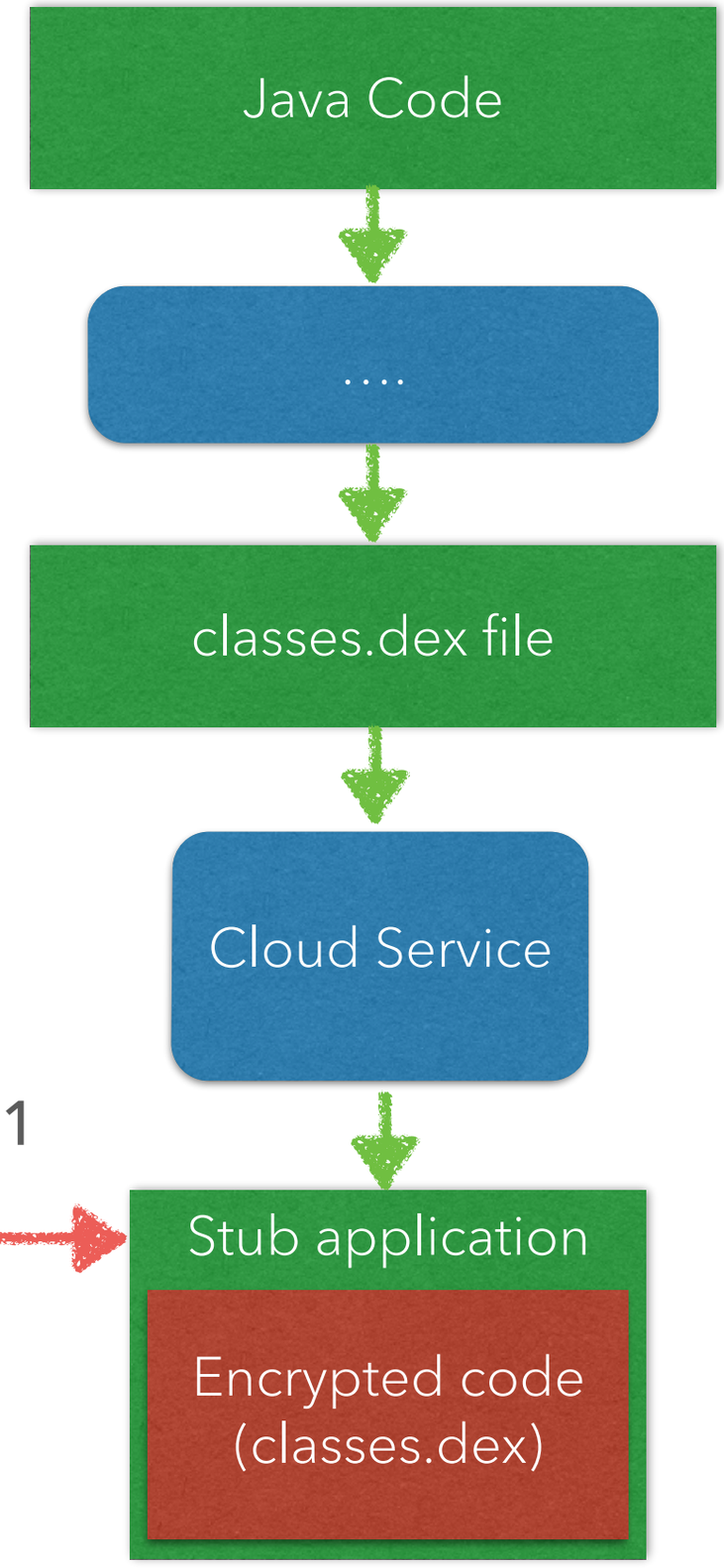
- Awesome concept and fun to reverse!
- Slight file size increase
- Prevents easily static analysis
- Interesting techniques to detect analysis (though not awesome)
- “Hard” once, easy afterwards
- Easily automated to unprotect
- Still has string encryption (similar to DexGuard/Allitori) afterwards
- Hacker Protection Factor 3

PACKERS

HOSEDEX2JAR

Packers

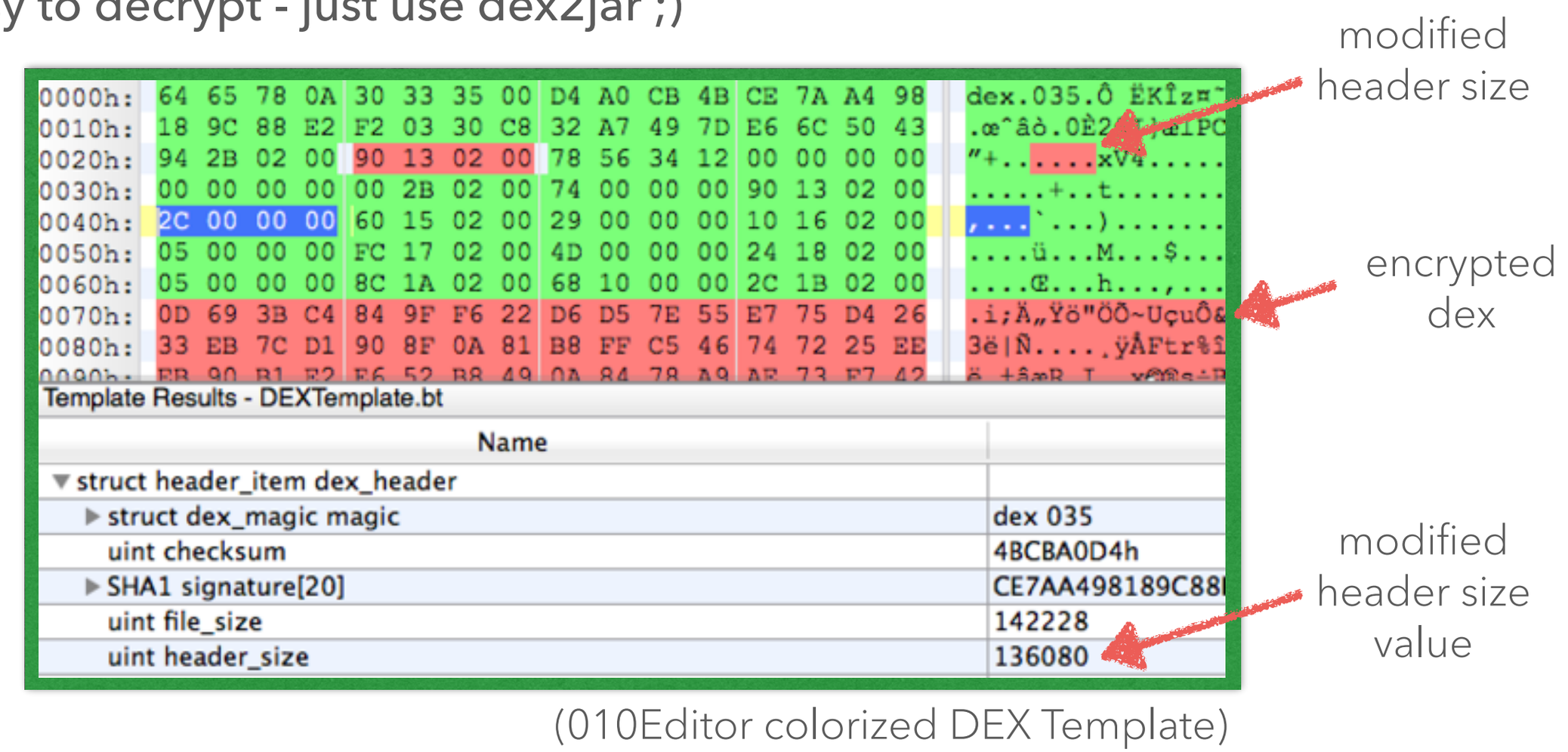
- “POC” Packer
- Not viable for real use
- Appears defunct
- Near zero ITW samples
- Mimics “Dexception” attack from Dex Education 101
- Cost: Free



HOSEDEX2JAR

Packers

- Encrypts and injects dex file into dex header (deception)
- Very easy to spot
- Very easy to decrypt - just use dex2jar ;)

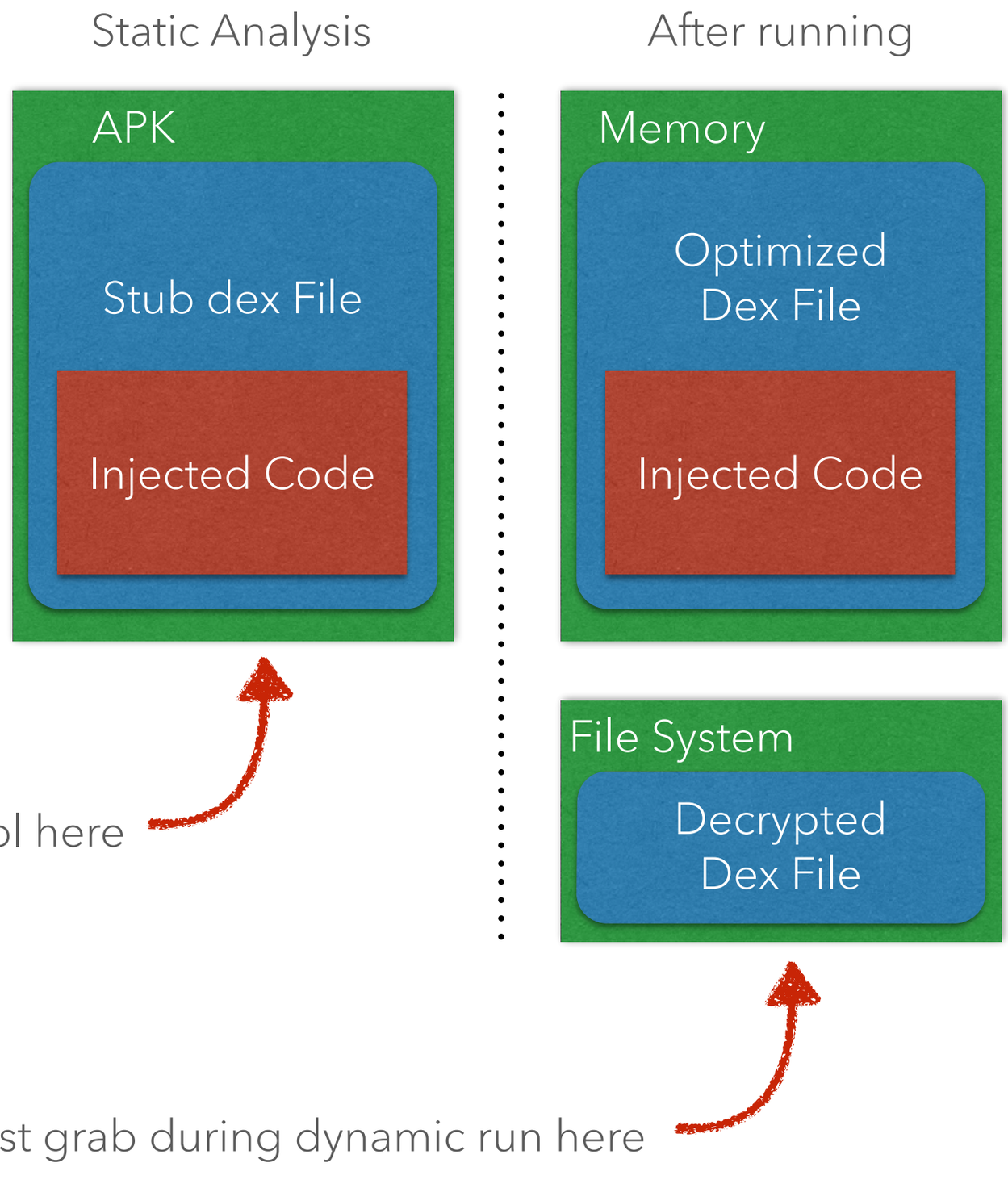


(010Editor colorized DEX Template)

HOSEDEX2JAR

Packers

- On execution loader stub decrypts in memory and dumps to file system
- Loader stub acts as proxy and passes events to the Dex file on file system using a DexClassLoader
- Static unpacker (wrapping stub code with dex2jar output) available; http://github.com/strazzere/dehoser/



HOSEDEX2JAR

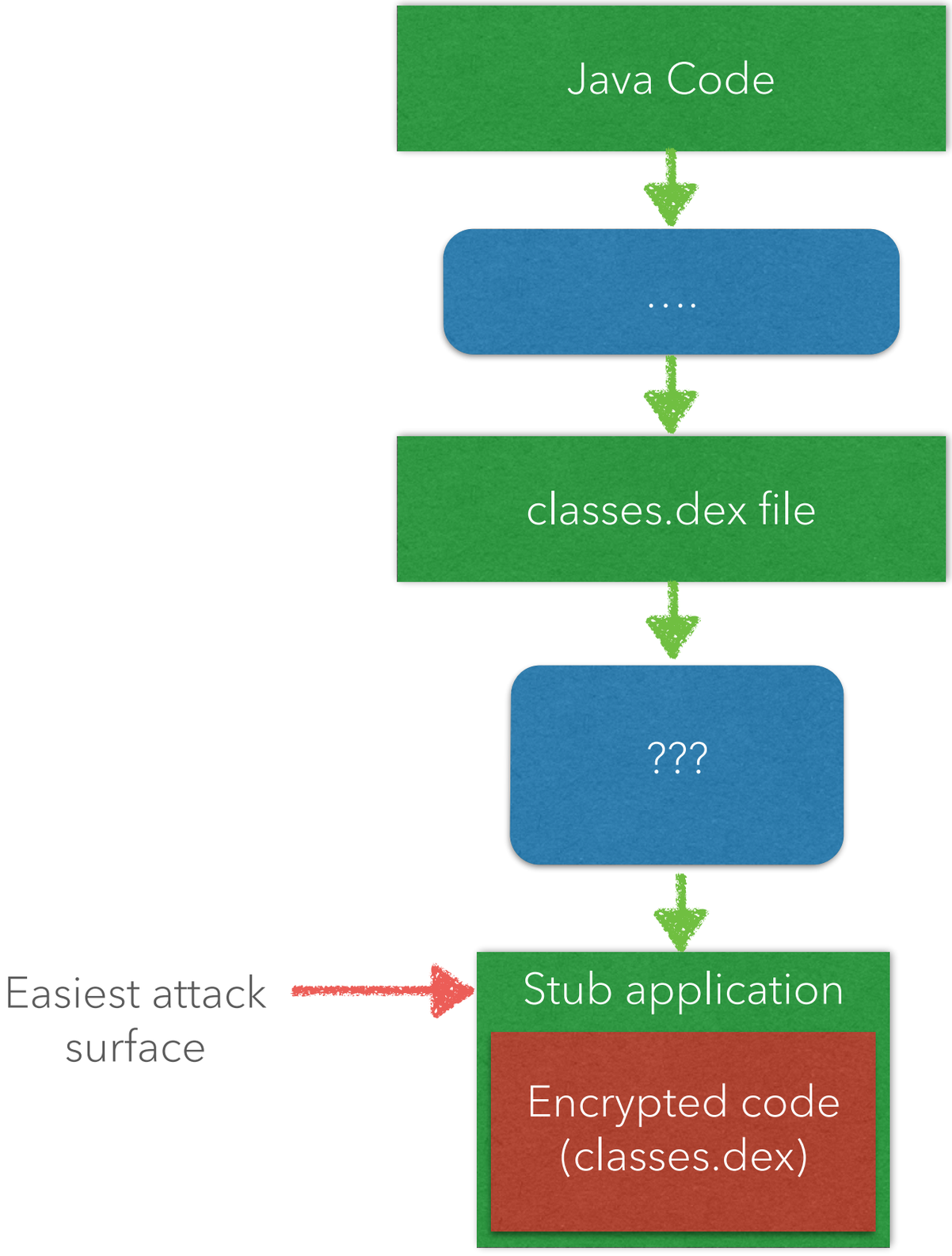
Packers

- Simple POC
- Slight file size increase
- Attempts to prevent static analysis - sort of works
- Lots of crashing
- Easily automated to unpack
- Easy to reverse, good for learning
- Hacker Protection Factor 0.5

PANGXIE

Packers

- Chinese Packer
- Anti-debug
- Anti-tamper
- ???
- Appears to be defunct product
- Little usage/samples ITW
- Cost: ???

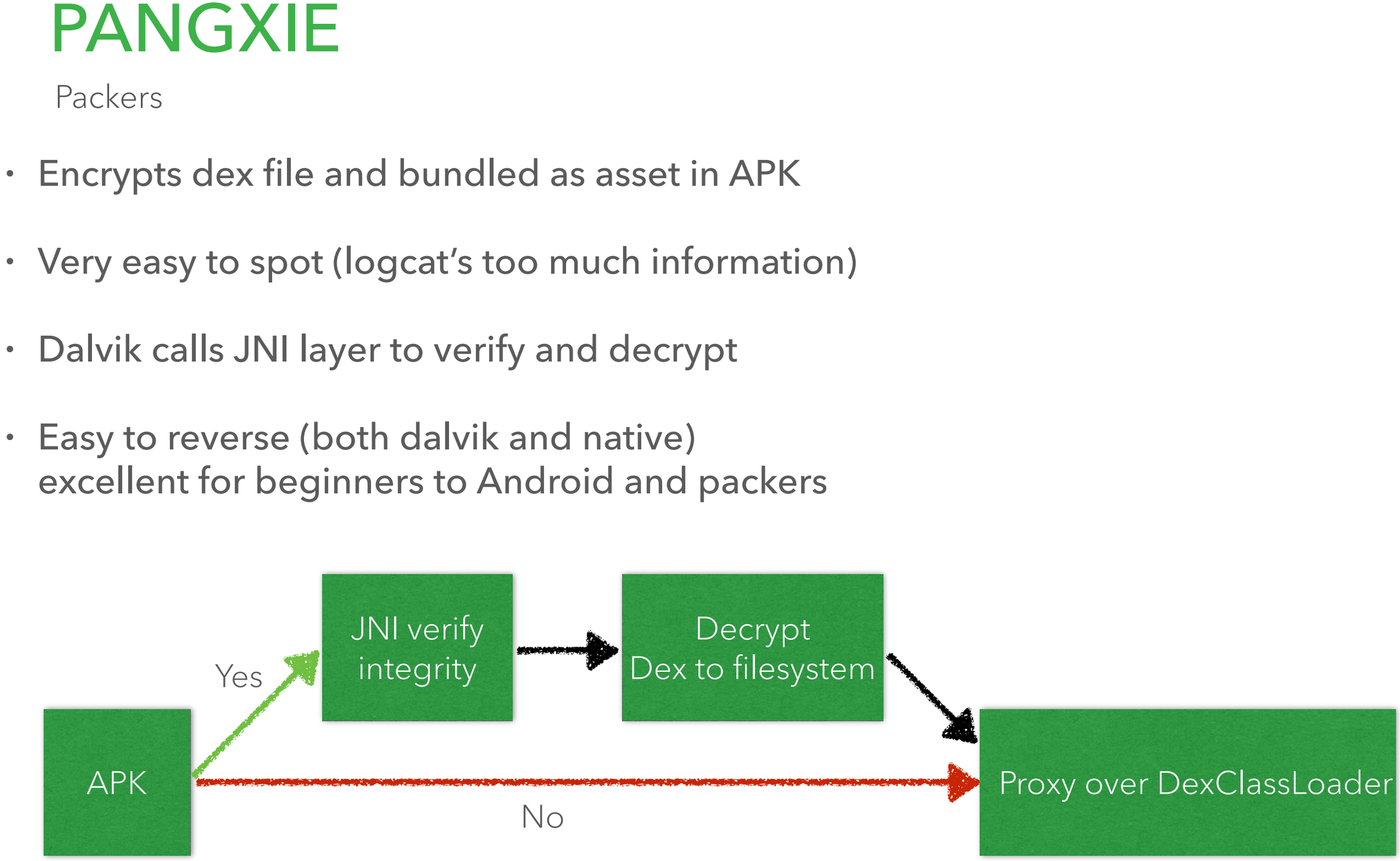


PANGXIE

Packers

- Encrypts dex file and bundled as asset in APK
- Very easy to spot (logcat's too much information)
- Dalvik calls JNI layer to verify and decrypt
- Easy to reverse (both dalvik and native)

excellent for beginners to Android and packers



PANGXIE

Packers

- AES “used” … only for digest verification
- Easily automated…

  0x54 always the “key”
- Or dynamically grab the /data/data/%package_name% /app_dex folder

PANGXIE

Packers

- Or dynamically grab the /data/data/%package_name%/app_dex folder

```
root@generic:/ # cd data/data/com.joytap.PetDash/
root@generic:/data/data/com.joytap.PetDash # ls -l
lrwxrwxrwx install  install          2014-07-24 00:08 lib -> /data/app-lib/com.joytap.PetDash-1
root@generic:/data/data/com.joytap.PetDash # ls -l
drwx------ u0_a48   u0_a48           2014-07-24 00:09 app_dex
drwxrwx--x u0_a48   u0_a48           2014-07-24 00:09 cache
lrwxrwxrwx install  install          2014-07-24 00:08 lib -> /data/app-lib/com.joytap.PetDash-1
drwxrwx--x u0_a48   u0_a48           2014-07-24 00:09 shared_prefs
root@generic:/data/data/com.joytap.PetDash # cd app_dex/
root@generic:/data/data/com.joytap.PetDash/app_dex # ls -l
-rw-r--r-- u0_a48   u0_a48    2463568 2014-07-24 00:09 classes.dex
-rw------- u0_a48   u0_a48     811496 2014-07-24 00:09 classes.jar
root@generic:/data/data/com.joytap.PetDash/app_dex #
```

PANGXIE

Packers

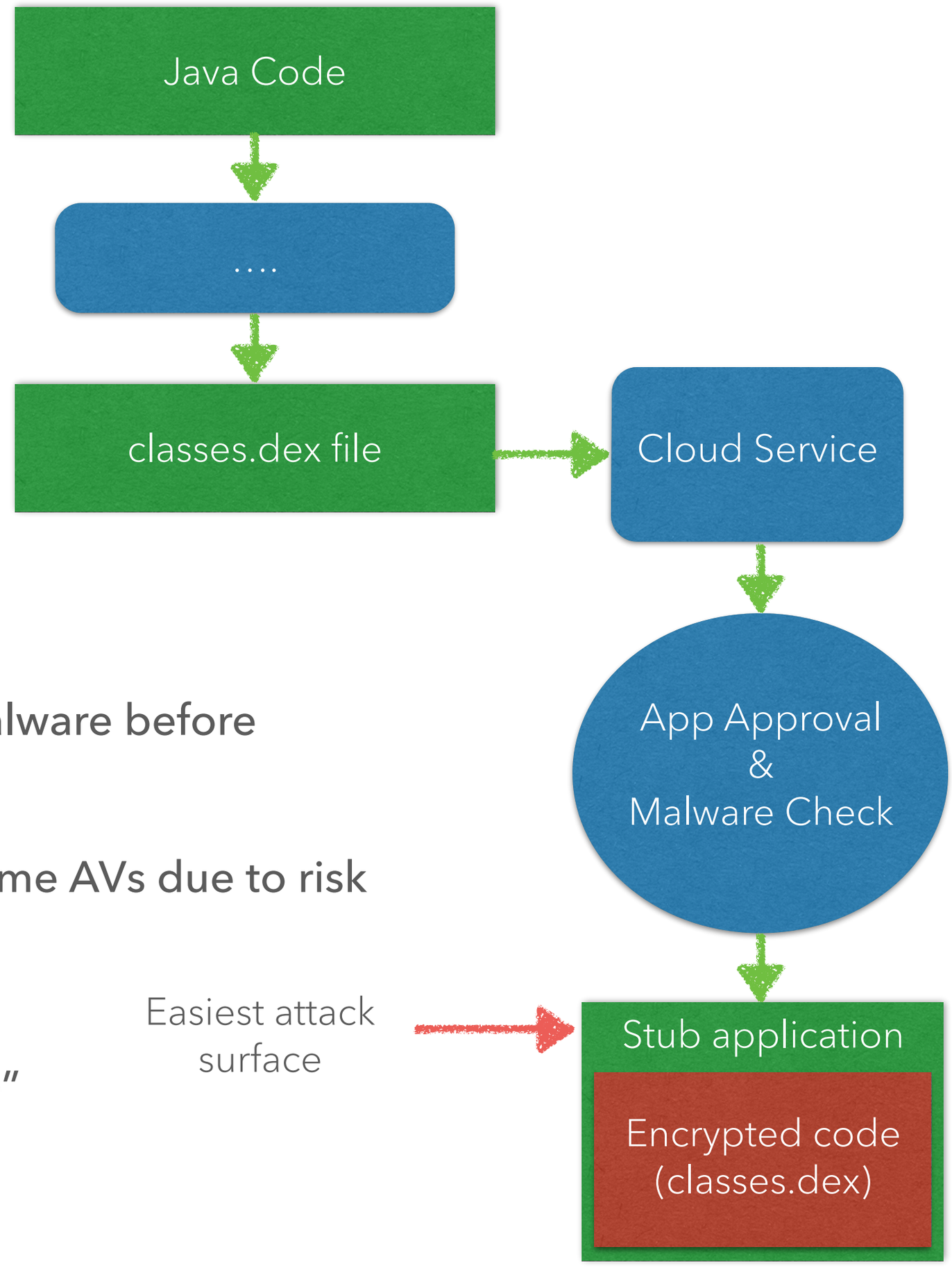
- Slight file size increase
- Prevents static analysis - though easy to identify
- Uses static 1 byte key for encryption
- Easily automated to unpack
- Very easy to reverse, good for learning
- Good example of an unobfuscated packer stub for cloning
- Hacker Protection Factor 1.5

BANGCLE

Packers

- Anti-debugging
- Anti-tamper
- Anti-decompilation
- Anti-runtime injection

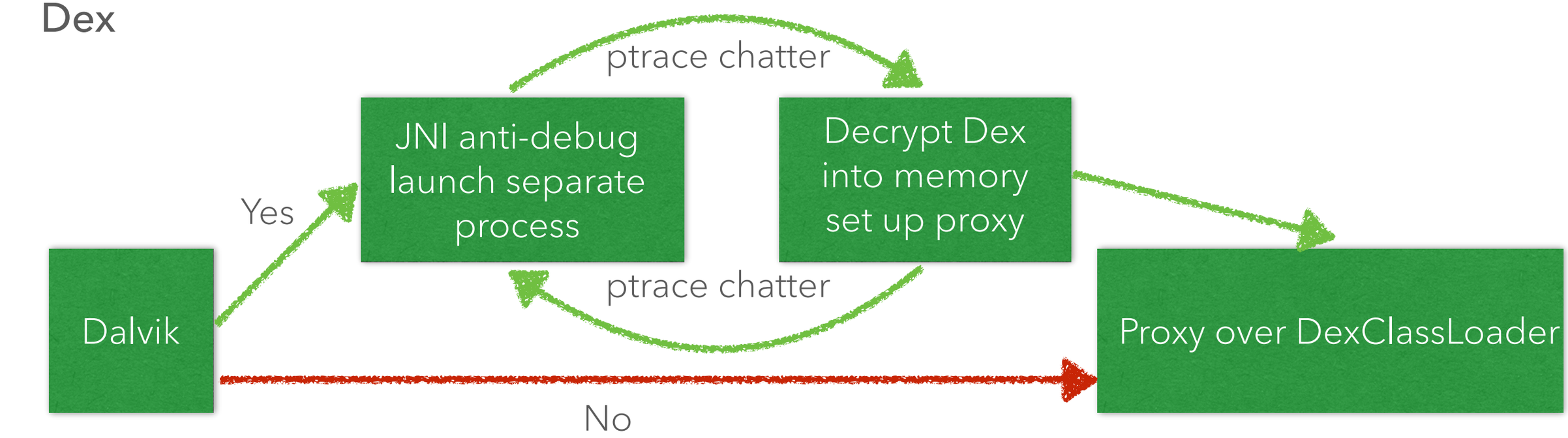
- Online only service
  - "APKs checked for malware before packaging"
- Generically detected by some AVs due to risk
- Cost: ~$10k
- "No one has done it before"



BANGCLE

Packers

- Dalvik execution talks launches JNI
- JNI launches a secondary process
- Chatter over PTRACE between the two processes
- Newest process decrypts Dex into memory
- Original Dalvik code proxies everything to decrypted Dex



BANGCLE

Packers

```
[86%]tstrazzere@bebop:[secapk] $ adb shell ps | grep com.gametowin.jiangshi
u0_a48    773   37    180620 35848 ffffffff 400433dc S com.gametowin.jiangshi
u0_a48    788   773   66700  1680  ffffffff 40034014 S com.gametowin.jiangshi
u0_a48    790   788   2016   380   c0091de4 40034014 S com.gametowin.jiangshi
```

Original Dalvik process

Two forked native processes

```
1|root@generic:/ # cat proc/773/cmdline
com.gametowin.jiangshiroot@generic:/ #
root@generic:/ # cat proc/788/cmdline
com.gametowin.jiangshicom.gametowin.jiangshi15382448160/data/app/com.gametowin.jiangshi-1.apk36com.gametowin.jiangshi4142
root@generic:/ # cat proc/790/cmdline
com.gametowin.jiangshicom.gametowin.jiangshi15382448160/data/app/com.gametowin.jiangshi-1.apk36com.gametowin.jiangshi4142
root@generic:/ # ls -l /proc/773/task/
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 773
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 775
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 777
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 778
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 780
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 781
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 782
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 783
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 784
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:28 785
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:31 786
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:31 789
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:31 793
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:31 794
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:31 803
dr-xr-xr-x u0_a48   u0_a48            2014-08-10 14:31 807
```

Cloned processes that are attachable

BANGCLE

Packers

```
1|root@generic:/ # cat /proc/807/maps
2a000000-2a002000 r-xp 00000000 1f:00 695        /system/bin/app_process
2a002000-2a003000 r--p 00001000 1f:00 695        /system/bin/app_process
2a003000-2a233000 rw-p 2a003000 00:00 0          [heap]
40000000-4000f000 r-xp 00000000 1f:00 703        /system/bin/linker
                                  .....
49cc4000-49cc5000 r--s 00040000 1f:01 626        /data/data/com.gametowin.jiangshi/.cache/classes.jar
49cc5000-49d6b000 rw-p 00000000 1f:01 642        /data/data/com.gametowin.jiangshi/.cache/classes.dex
49d6b000-49d6c000 rw-p 49d6b000 00:00 0
49d6c000-49e12000 r--p 49d6c000 00:00 0
49e12000-49e1c000 rw-p 00000000 00:07 2168       /dev/ashmem/dalvik-aux-structure (deleted)
```

Always the decrypted memory region

Still encrypted

```
root@generic:/ # ./data/local/tmp/kisskiss com.gametowin.jiangshi
[*] Android Dalvik Unpacker/Unprotector - <diff@lookout.com>
 [+] Hunting for com.gametowin.jiangshi
 [+] 773 is service pid
 [+] 807 is clone pid
 [+] Attempting to detect packer/protector...
  [*] Nothing special found, assuming Bangcle...
 [+] Unpacked odex found in memory!
 [+] Attempting to dump memory region 0x49d6c000 to 0x49e12000
 [+] Unpacked/protected file dumped to : /data/local/tmp/com.gametowin.jiangshi.dumped_odex
```

BANGCLE

Packers

- Well written, lots of anti-* tricks
- Seems to be well supported and active on development
- Does a decent job at online screening - no tool released for download
    - Though things clearly to slip through
- Not impossible to reverse and re-bundle packages
- Current weakness (for easy runtime unpacking) is having a predictable unpacked memory location
- Hacker Protect Factor 5

NOW WHAT?

CODE!

- Open-sourced unpacker
  - https://github.com/strazzere/android-unpacker (push after this talk)
  - Bangcle
    - Most popular/highest prevalence
    - Plenty of malicious/grey area samples
  - APKProtect
    - High prevalence and graining more traction (offline tools)
    - Malicious/grey area samples
  - More packers added as malware/prevalence emerges

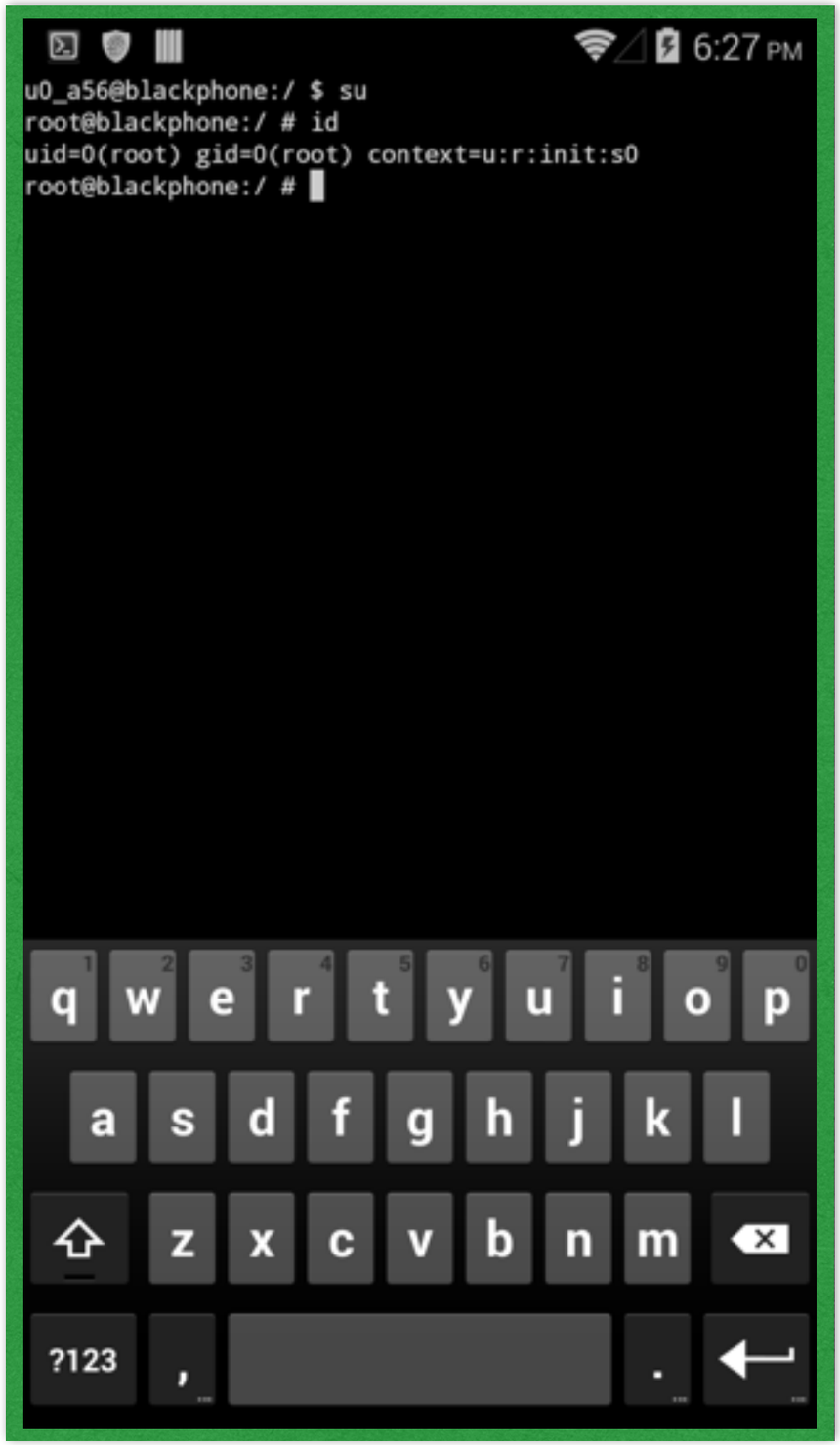
- Slim anti-detection code
  - APKProtect LD_PRELOAD module (same repo as android-unpacker)
  - https://github.com/strazzere/android-lkms

- Malicious samples uploaded soon to ContagioMinidump (mobile malware)
  - http://contagiominidump.blogspot.com/

BLACKPHONE

What you're actually here for…

- ROOTED!
- Three stages of exploits
- Requires user interaction



BLACKPHONE

Enabled ADB

- Stage 1

- “turned ADB off because it causes a software bug and potentially impacts the user experience"

- Removed UI accessibility from settings APK

- Just send an intent to pop the menu

```
ComponentName intentComponent = new ComponentName("com.android.settings",
                "com.android.settings.Settings$DevelopmentSettingsActivity");
Intent mainIntent = new Intent("android.intent.action.MAIN");
mainIntent.setComponent(intentComponent);
startActivity(mainIntent);
```

BLACKPHONE

Get System UID

- Stage 2
- Fixed in latest OTA (vuln out of box though)
- System privledged APK w/ debuggable set to true
- Allows us to get System UID
- Enlarge attack surface
- http://www.saurik.com/id/17 exploit how-to

```
[51%]tstrazzere@bebop:[wipe] $ axml AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:sharedUserId="android.uid.system"
        android:versionCode="6"
        android:versionName="0.8.2"
        package="com.karumi.blackphone.wipe"
        >
```

```
<application
        android:theme="@7F0C0000"
        android:label="@7F0B0018"
        android:icon="@7F020072"
        android:debuggable="true"
        android:allowBackup="false"
        >
```

BLACKPHONE

System to root

- Stage 3
- There are some out there for Android
- One has been used here
- Sorry - cannot currently disclose!

BLACKPHONE

DEMO

- Stage 1 - Enable ADB
- Stage 2 - Get System UID
- Stage 3 - System to root

THANKS!

TIM “DIFF” STRAZZERE
@TIMSTRAZZ

JON “JUSTIN CASE” SAWYER
@TEAMANDIRC

Join use on Freenode on #droidsec

Good people to follow on twitter for
Android/reversing/malware/hacking information;

@jduck @Fuzion24 @Gunther_AR @caleb_fenton @thomas_cannon
@droidsec @marcwrogers @osxreverser @cryptax @pof @quine
@0xroot @Xylitol @djbliss @saurik @collinrm @snare
#MalwareMustDie

08.10.2014

Defcon 22

APPLIED CYBERSECURITY
LLC