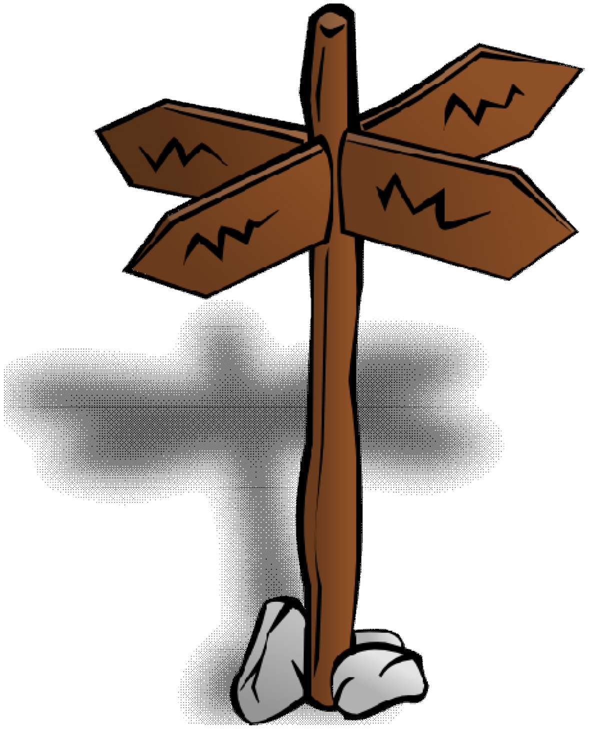
Reverse Engineering Android: Disassembly & Code Injection

Thanasis Petsas

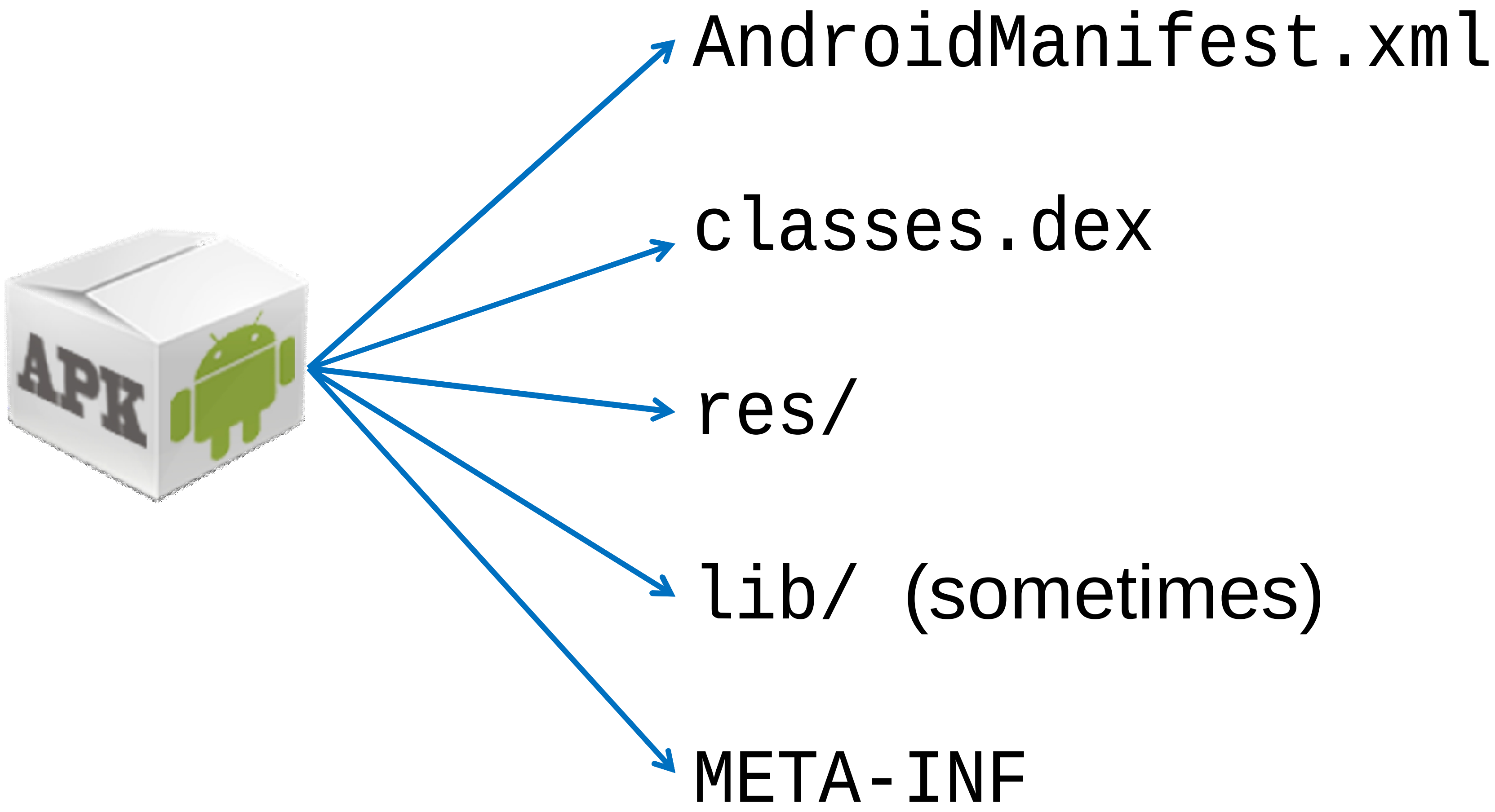
petsas@ics.forth.gr

Roadmap

- The APK Structure
- The Tools
- Hacking Approach
- Disassembly & App Analysis
- Code Injection



The APK Structure



AndroidManifest.xml

classes.dex

res/

lib/ (sometimes)

META-INF

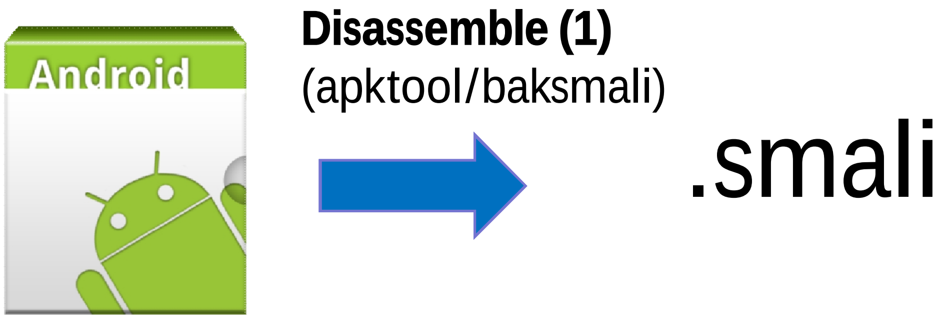
The Tools

- You'll need…
  - Android SDK
  - apktool (based on Smali/Baksmali)
  - jarsigner
  - keytool

Hacking Approach

Hacking Approach

1. Unzip APK & disassemble classes.dex

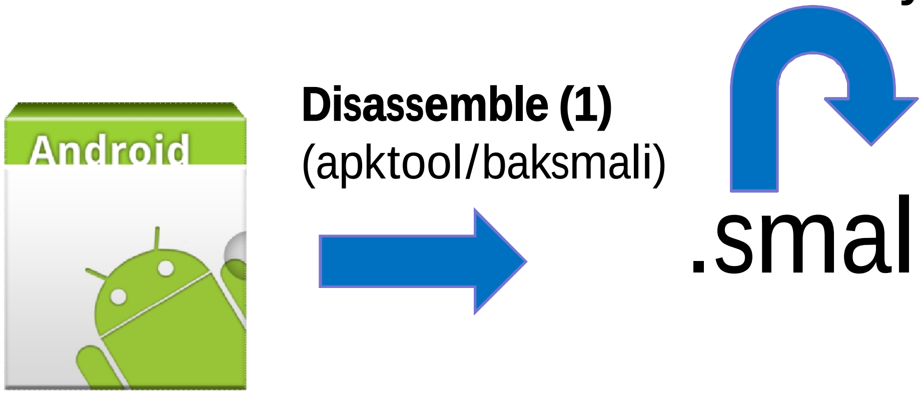


Disassemble (1)
(apktool/baksmali)

.smali

Hacking Approach

1. Unzip APK & disassemble classes.dex
2. Perform static analysis on the app



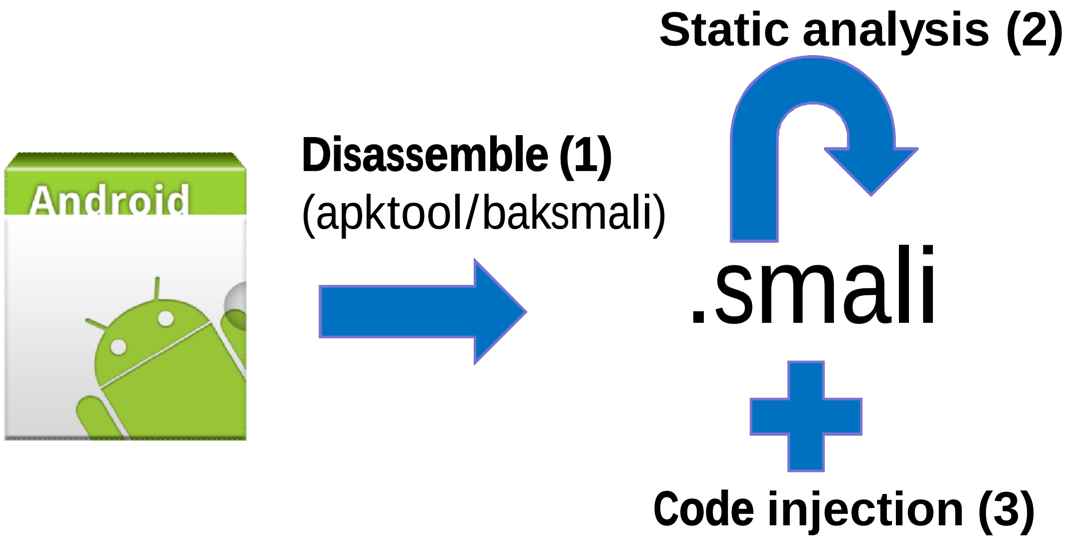
Static analysis (2)

Disassemble (1)
(apktool/baksmali)

.smali

Hacking Approach

1. Unzip APK & disassemble classes.dex
2. Perform static analysis on the app
3. Inject byte-code into the app



Static analysis (2)

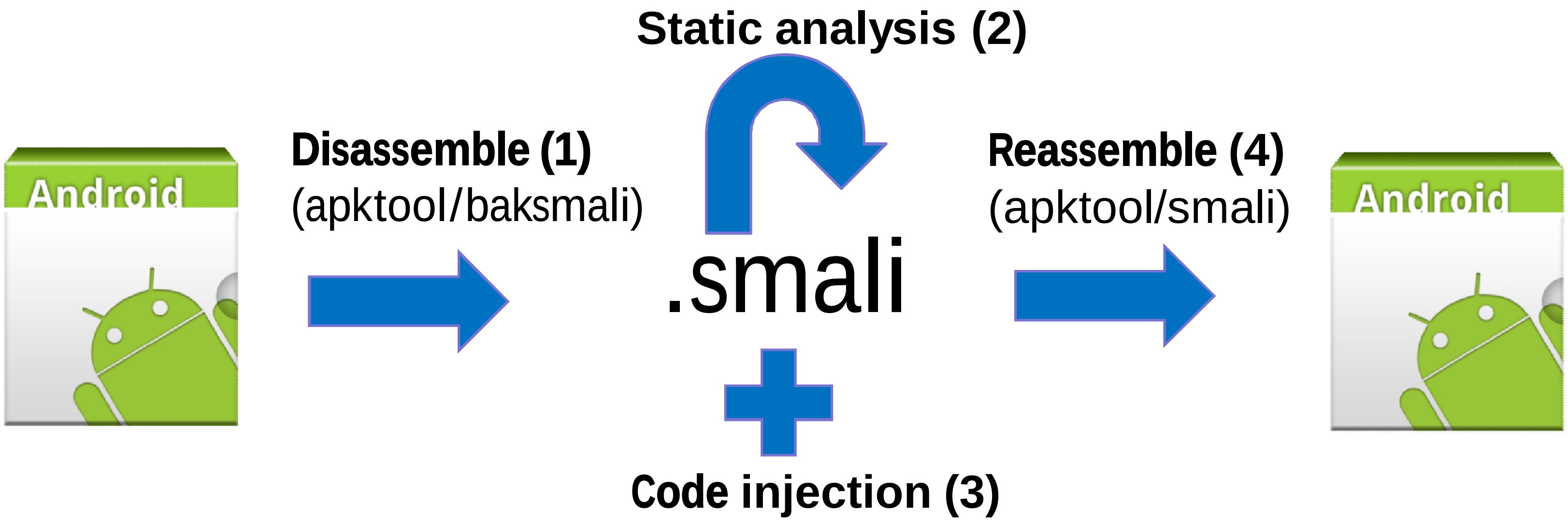
Disassemble (1)
(apktool/baksmali)

.smali

Code injection (3)

Hacking Approach

1. Unzip APK & disassemble classes.dex
2. Perform static analysis on the app
3. Inject byte-code into the app
4. Reassemble classes.dex & zip/sign APK



Static analysis (2)

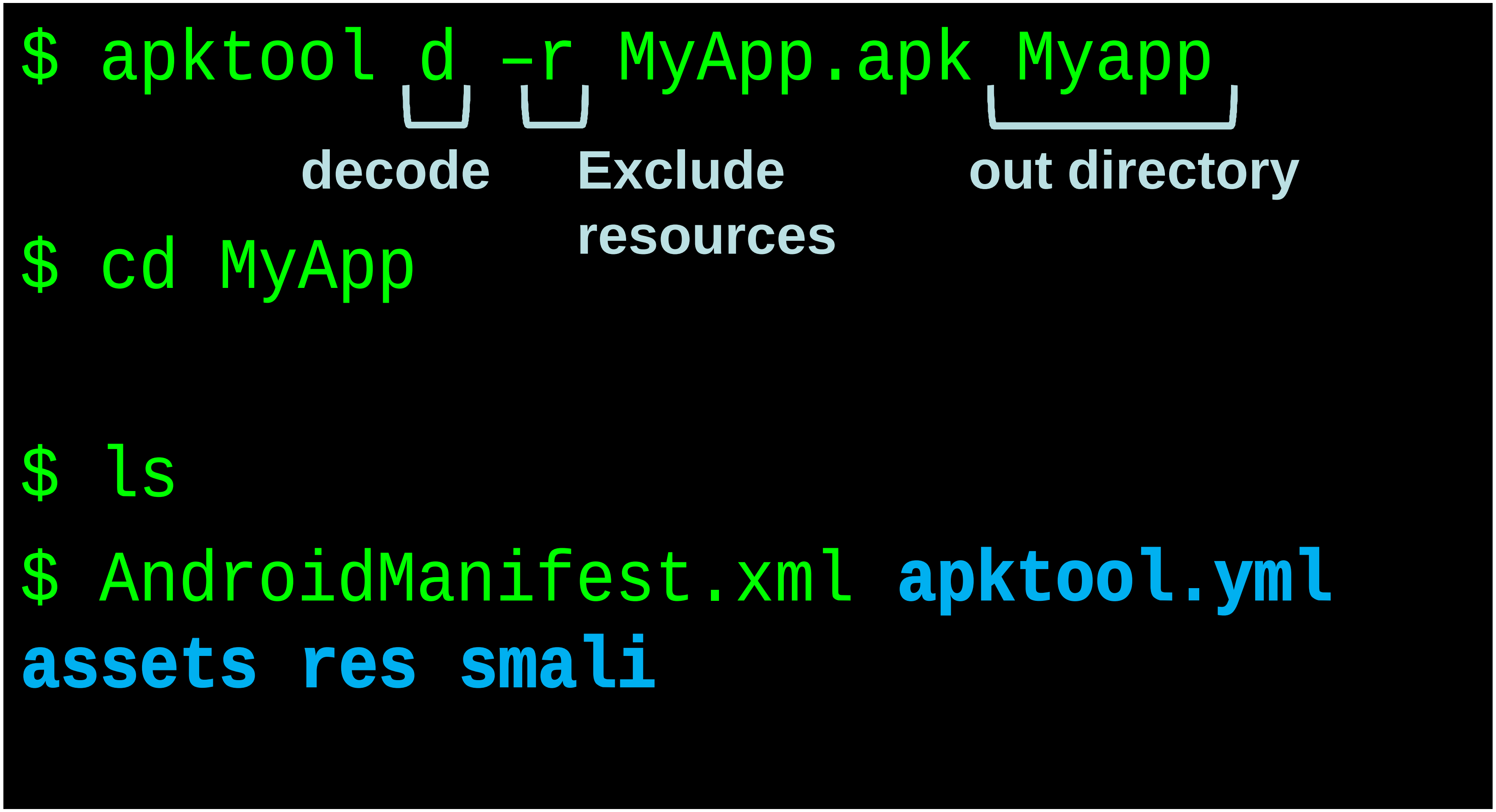
Disassemble (1)
(apktool/baksmali)

.smali

Reassemble (4)
(apktool/smali)

Code injection (3)

Disassembling APK



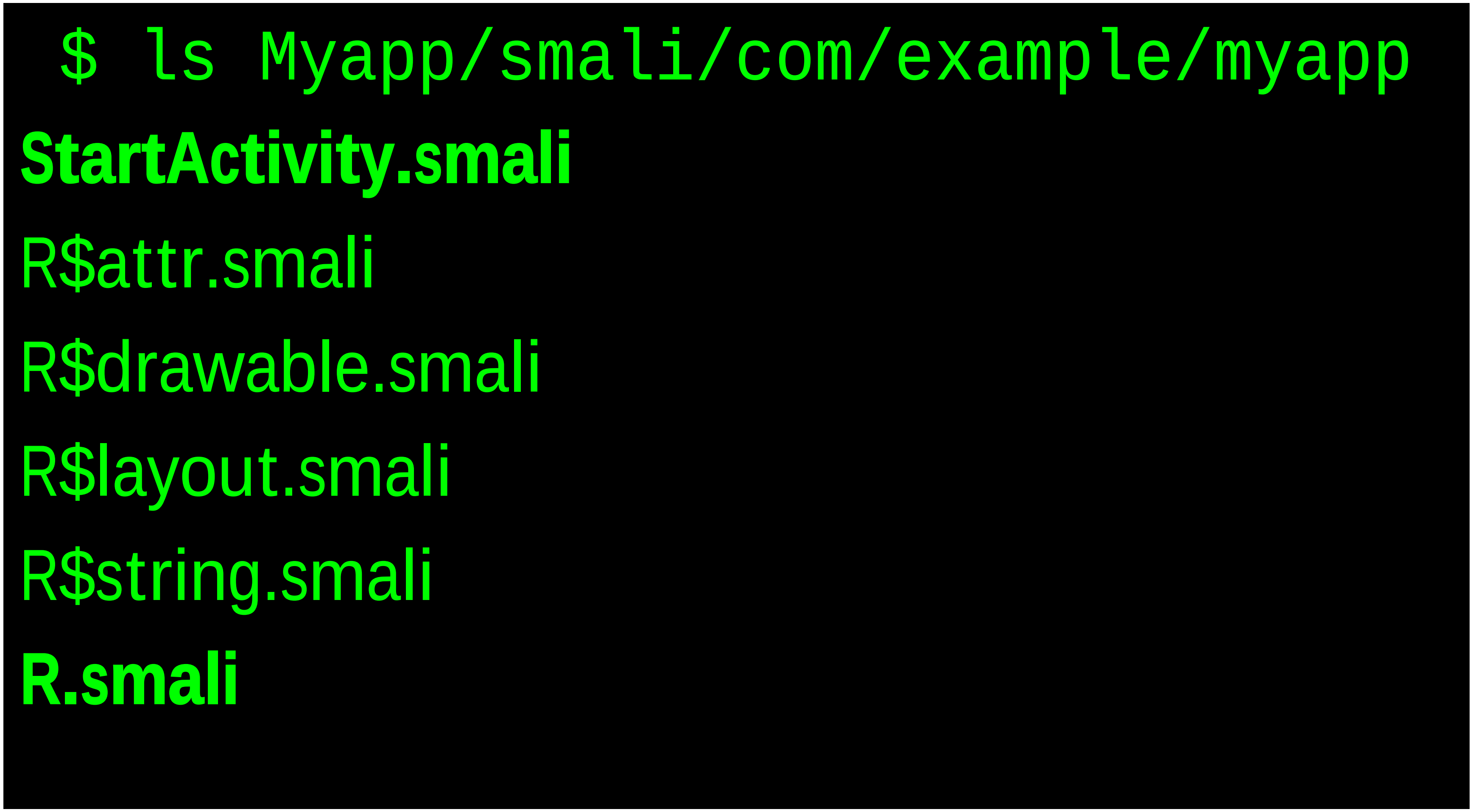
```
$ apktool d -r MyApp.apk Myapp

$ cd MyApp

$ ls
$ AndroidManifest.xml apktool.yml
assets res smali
```

decode

Exclude resources

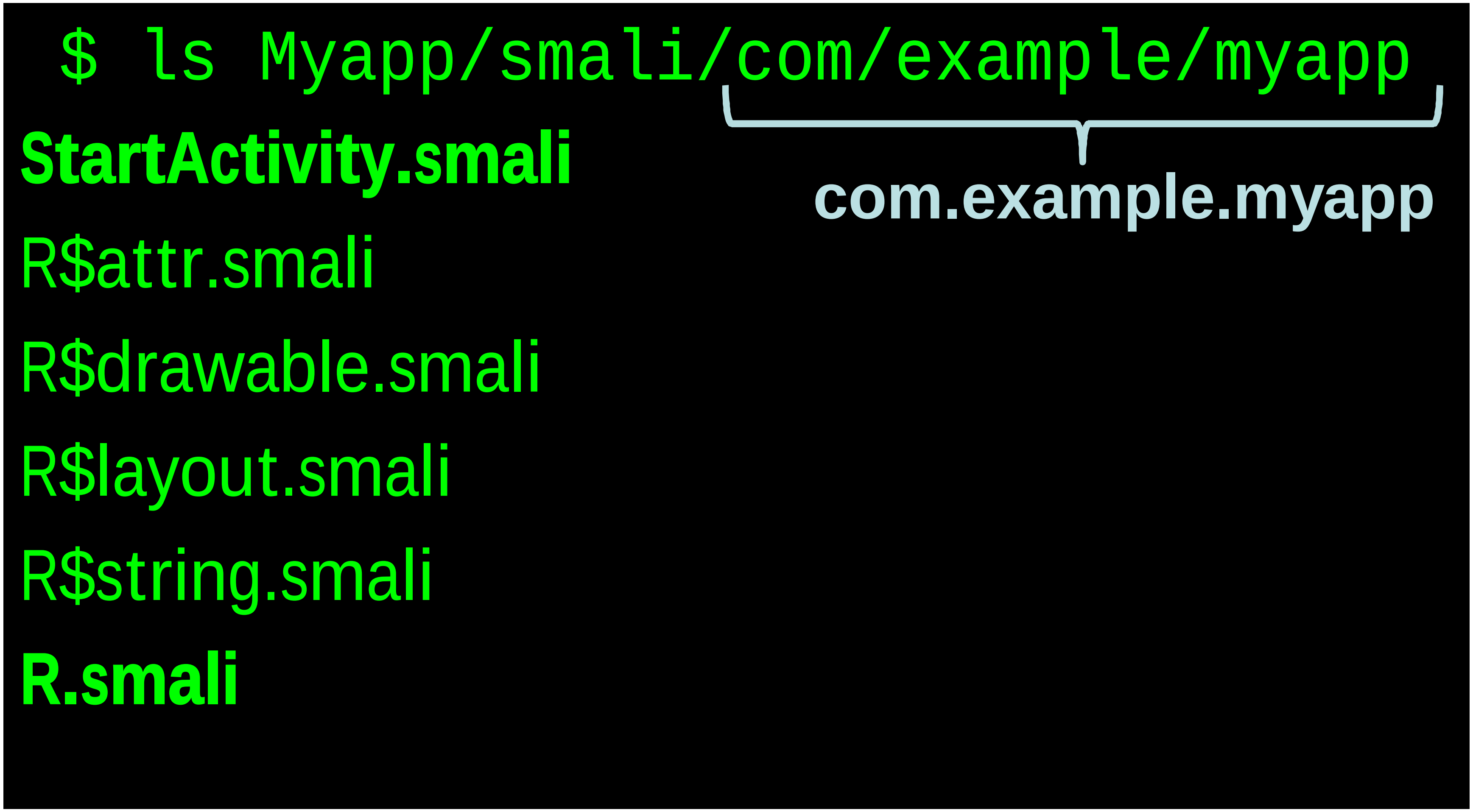
out directory

Analyzing the APK



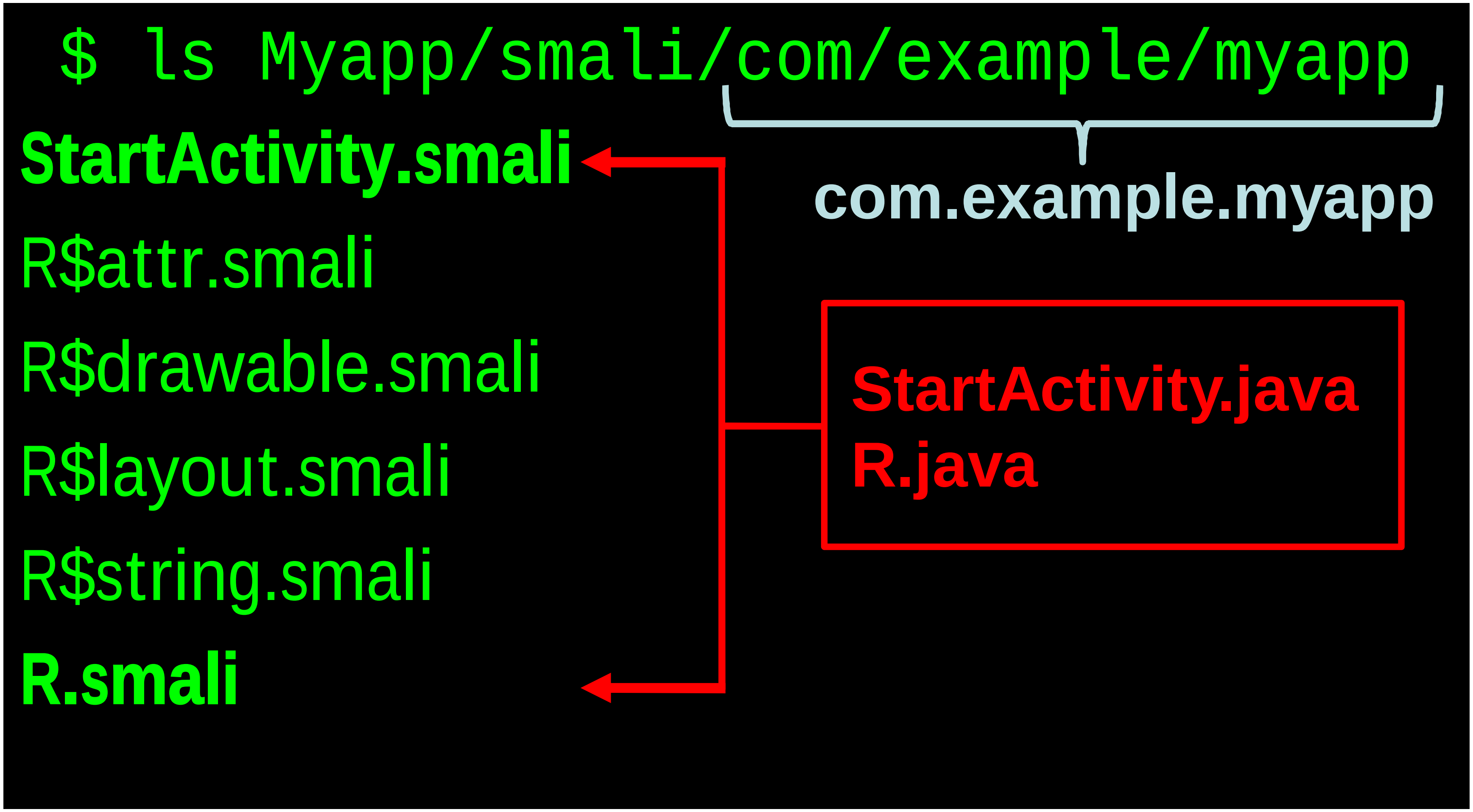
```
 $ ls Myapp/smali/com/example/myapp
StartActivity.smali
R$attr.smali
R$drawable.smali
R$layout.smali
R$string.smali
R.smali
```

Analyzing the APK



```
 $ ls Myapp/smali/com/example/myapp
StartActivity.smali
R$attr.smali
R$drawable.smali
R$layout.smali
R$string.smali
R.smali
```

Analyzing the APK



```
$ ls Myapp/smali/com/example/myapp
StartActivity.smali
R$attr.smali
R$drawable.smali
R$layout.smali
R$string.smali
R.smali
```

com.example.myapp

StartActivity.java
R.java

Java to Smali

```
…

public class StartActivity extends Activity {

  @Override
   protected void onCreate(
          Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_start);

      Log.i("StartActivity:", “Message");

}
```

Java code

```
…

# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 3
    .parameter "savedInstanceState"
    .prologue
    invoke-super {p0, p1}, Landroid/app/Activity
     ;->onCreate(Landroid/os/Bundle;)V

     const/high16 v0, 0x7f03

const-string v0, "StartActivity:"
const-string v1, “Message"
 invoke-static {v0, v1}, Landroid/util/Log;
        ->d(Ljava/lang/String;Ljava/lang/String;)I
 move-result v0

    return-void
.end method
```

Smali Byte code

Java to Smali

```
…

public class StartActivity extends Activity {

  @Override
   protected void onCreate(
          Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_start);

     Log.i("StartActivity:", “Message");

}
```

Java code

```
…

# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 3
    .parameter "savedInstanceState"
    .prologue
    invoke-super {p0, p1}, Landroid/app/Activity
     ;->onCreate(Landroid/os/Bundle;)V

     const/high16 v0, 0x7f03

const-string v0, "StartActivity:"
const-string v1, “Message"
 invoke-static {v0, v1}, Landroid/util/Log;
        ->d(Ljava/lang/String;Ljava/lang/String;)I
 move-result v0

    return-void
.end method
```

Smali Byte code

Java to Smali

```
…

public class StartActivity extends Activity {

  @Override
  protected void onCreate(
        Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_start);

    Log.i("StartActivity:", “Message");

}
```

Java code

```
…

# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 3
    .parameter "savedInstanceState"
    .prologue
    invoke-super {p0, p1}, Landroid/app/Activity
     ;->onCreate(Landroid/os/Bundle;)V

     const/high16 v0, 0x7f03

const-string v0, "StartActivity:"
const-string v1, “Message"
 invoke-static {v0, v1}, Landroid/util/Log;
        ->d(Ljava/lang/String;Ljava/lang/String;)I
 move-result v0

    return-void
.end method
```

Smali Byte code

Class Representation in Smali

```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java”

# static fields
.field public static final TAG:Ljava/lang/String; = “String”

# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```

Class information

Static fields

Methods
Direct
Virtual

Class Representation in Smali

[redacted]

Class information

```
# static fields
.field public static final TAG:Ljava/lang/String; = “String”
```

Static fields

```
# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```

Methods
Direct
Virtual

Class Representation in Smali

```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java”

# static fields
.field public static final TAG:Ljava/lang/String; = “String”

# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```

Class information

Static fields

Methods
Direct
Virtual

Class Representation in Smali

```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java”
```

Class information

[redacted]

Static fields

```
# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```

Methods
Direct
Virtual

Class Representation in Smali

```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java”

# static fields
.field public static final TAG:Ljava/lang/String; = “String”

# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```

Class information

Static fields

Methods
Direct
Virtual

Class Representation in Smali

```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java”

# static fields
.field public static final TAG:Ljava/lang/String; = “String”

# direct methods
```

Class information

Static fields

Methods
Direct
Virtual

Class Representation in Smali

```
.class public Lcom/apkudo/util/Serializer;
.super Ljava/lang/Object;
.source "Serializer.java”

# static fields
.field public static final TAG:Ljava/lang/String; = “String”

# direct methods
.method public constructor <init>()V
    .registers 1

    .prologue
    .line 5
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V

    return-void
.end method
```

Class information

Static fields

Methods
Direct
Virtual

Smali Syntax – Types

```
.method private doSomething()V
```

V	void
Z	boolean
B	byte
S	short
C	char
F	float
I	int
J	long
D	double
[	array

Smali Syntax – Classes

```
Lcom/example/myapp/MyClass;
```

- full name space slash separated
- prefixed with L
- suffixed with ;

```
StringBuilder sb = new StringBuilder(“str”)
```

```
new-instance v1, Ljava/lang/StringBuilder;
const-string v2, “str"

invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-
><init>(Ljava/lang/String;)V
```

Smali Syntax – Methods

```
.method private doSomething()V
```

keyword method name parameters/return

```
.method private delayedAnimationFrame(J)Z
    .registers 8
    .parameter "currentTime”
```

```
# Static invocation
invoke-static {p2}, Landroid/text/TextUtils;
                   ->isEmpty(Ljava/lang/CharSequence;)Z

# Virtual invocation
invoke-virtual {v0, v1}, Lcom/google/android/finsky/FinskyApp;
                                             ->drainAllRequests(I)V
```

Smali Syntax – Registers

- `.locals` → # registers of a method without parameters
- #parameters → # input parameters + (`p0`: `this` reference)

```
v0 - local 0
p0 - parameter 0 (this)
p1 - parameter 1
```

Smali Syntax – Opcodes

- `invoke-super vx, vy, …`
- `new-instance vx`
- `invoke-direct vx, vy, …`
- `const-string vx`
- `invoke-virtual vx, vy, …`
- `return-void`

Hacking the App

- Let’s inject some code in the APK:
  - A toast message “hacked!”

    Java code:

    ```
    Toast.makeText(getApplicationContext(),
     "Hacked!", Toast.LENGTH_SHORT).show();
    ```

- How do we do this in smali?

→ Easy, let’s just **compile** this **into another app** (*e.g.*, MyApp2) and disassemble

Result

```
Toast.makeText(getApplicationContext(),
"Hacked!", Toast.LENGTH_SHORT).show();
```

Java code

```
invoke-virtual {p0}, Lcom/example/myapp2/TestActivity;
->getApplicationContext()Landroid/content/Context;
move-result-object v1
const-string v2, "Hacked!“
const/4 v3, 0x0
invoke-static {v1, v2, v3}, Landroid/widget/Toast;
->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)
Landroid/widget/Toast;
move-result-object v1
invoke-virtual {v1}, Landroid/widget/Toast;->show()V
```

Smali Byte code

Result

```
Toast.makeText(getApplicationContext(),
"Hacked!", Toast.LENGTH_SHORT).show();
```

Java code

```
invoke-virtual {p0}, Lcom/example/myapp2/TestActivity;
->getApplicationContext()Landroid/content/Context;
move-result-object v1
const-string v2, "Hacked!“
const/4 v3, 0x0
invoke-static {v1, v2, v3}, Landroid/widget/Toast;
->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)
Landroid/widget/Toast;
move-result-object v1
invoke-virtual {v1}, Landroid/widget/Toast;->show()V
```

Smali Byte code

Result

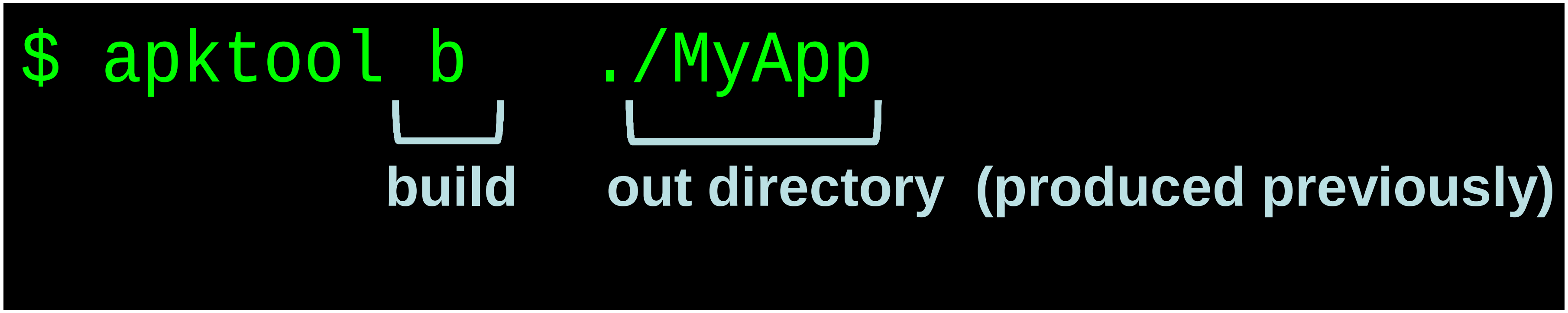
```
Toast.makeText(getApplicationContext(),
"Hacked!", Toast.LENGTH_SHORT).show();
```

Java code

```
invoke-virtual {p0}, Lcom/example/myapp2/TestActivity;
->getApplicationContext()Landroid/content/Context;
move-result-object v1
const-string v2, "Hacked!“
const/4 v3, 0x0
invoke-static {v1, v2, v3}, Landroid/widget/Toast;
->makeText(Landroid/content/Context;Ljava/lang/CharSequence;I)
Landroid/widget/Toast;
move-result-object v1
invoke-virtual {v1}, Landroid/widget/Toast;->show()V
```

Smali Byte code

Rebuilding the APK



```
$ apktool b   ./MyApp
```

build out directory (produced previously)

- This will instruct apktool to rebuild the app
- The path to the new APK: `./Myapp/`**`dist`**`/Myapp.apk`
- But this app is **not yet signed**

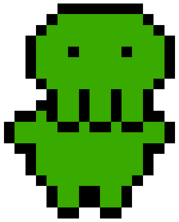
Signing the APK

```
$ keytool -genkey -v -keystore my-release-
key.keystore -alias alias_name -keyalg RSA
-validity 10000


$ jarsigner -verbose -sigalg MD5withRSA -
digestalg SHA1 -keystore my-release-
key.keystore ./MyApp/dist/MyApp.apk
alias_name
```

Installing the APK

```
# remove it first, if it is already
installed using its package name
$ adb uninstall com.example.myapp


# then, install it
$ adb install ./MyApp/dist/MyApp.apk
```

Practical Session



- You have received a malicious app named **Cthulhu.apk**
- This app sends some sensitive information to a malicious server
- Enhance the app with a **static evasion heuristic** so that it will expose its malicious activity only when running on a device
- You don't have access to the app's source code

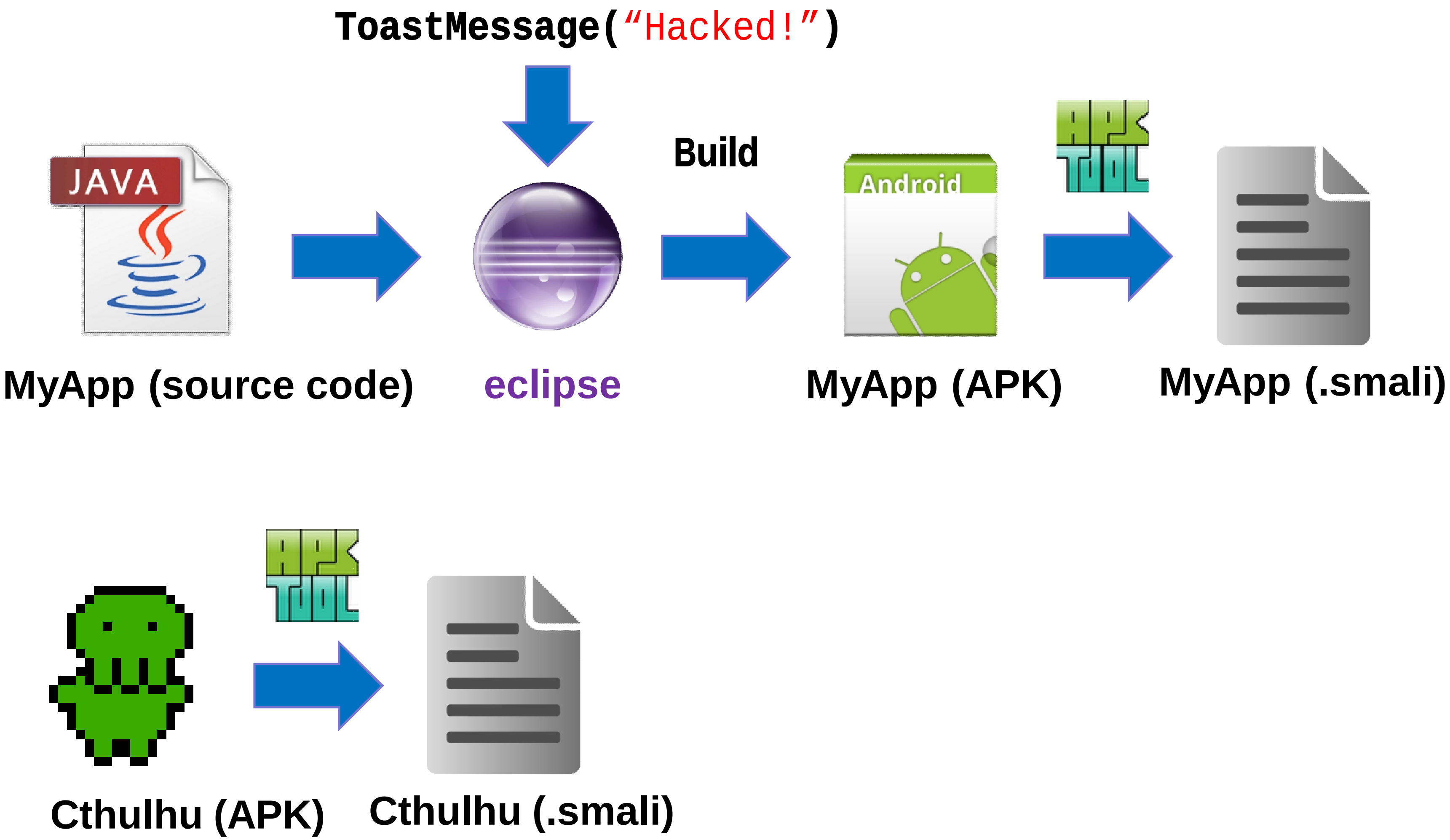
Steps to follow

a. First make Cthulhu app to display a toast message **"hacked!"** (hint: use MyApp)

b. Patch Cthulhu app with the evasion heuristic – IMEI check (hint: use again MyApp)

c. Submit the app to an online analysis service (*e.g.*, Andrubis)

Step a. – Inject a Toast Message



`ToastMessage(“Hacked!”)`

Build

MyApp (source code)

eclipse

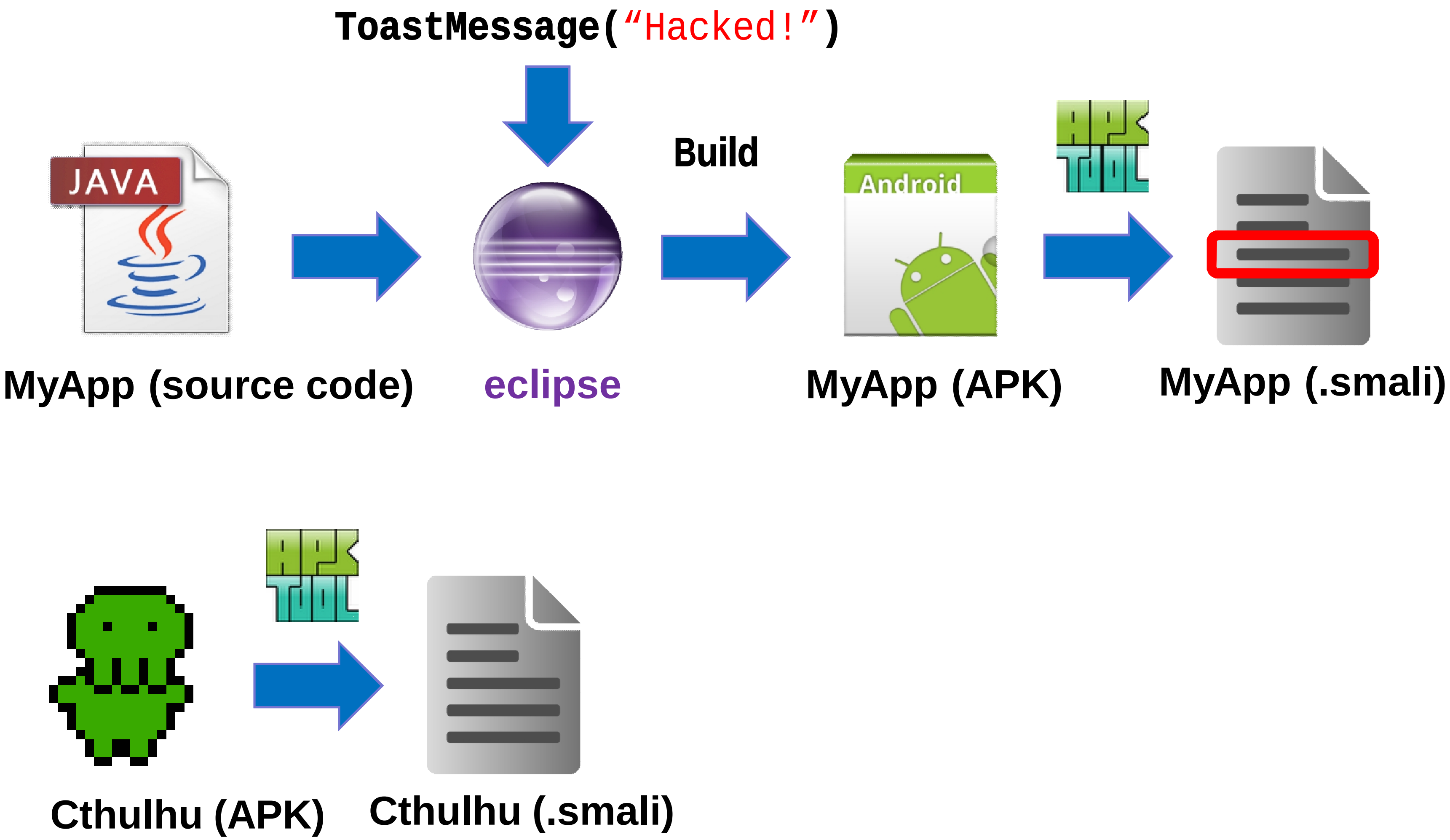
MyApp (APK)

MyApp (.smali)

Cthulhu (APK)

Cthulhu (.smali)

Step a. – Inject a Toast Message



ToastMessage("Hacked!")

Build

MyApp (source code)

eclipse

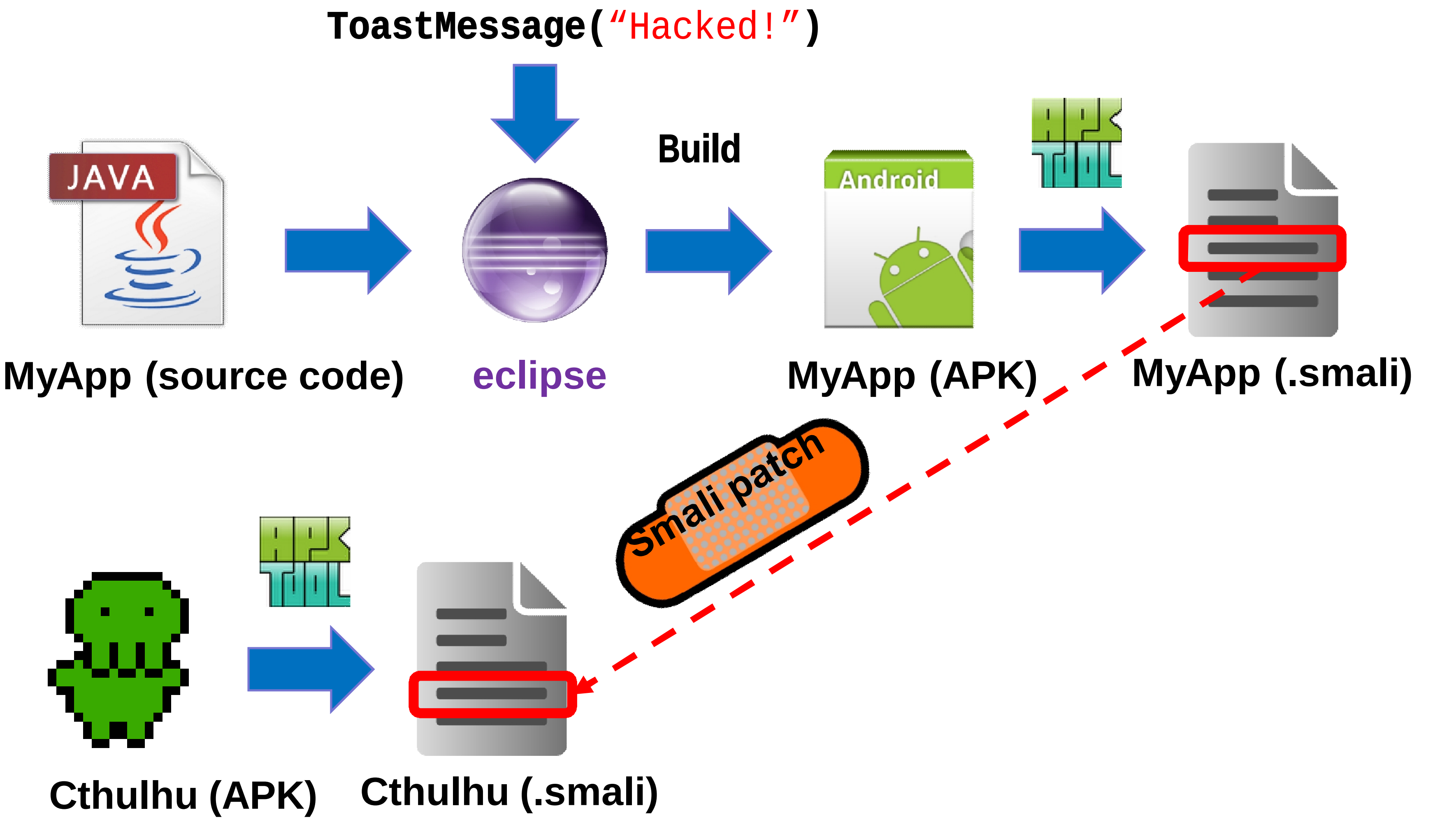
MyApp (APK)

MyApp (.smali)

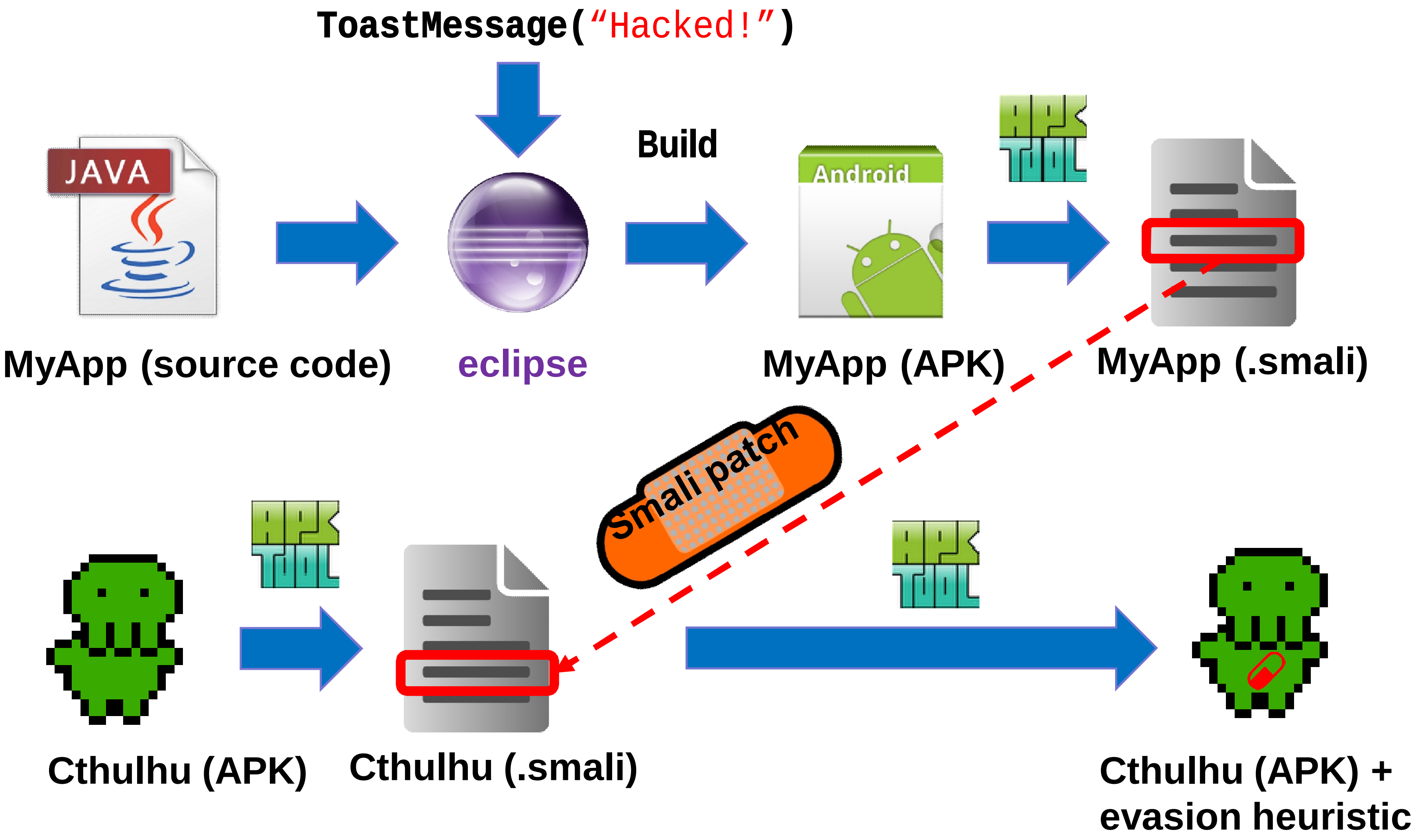
Cthulhu (APK)

Cthulhu (.smali)

Step a. – Inject a Toast Message



Step a. – Inject a Toast Message



`ToastMessage("Hacked!")`

Build

MyApp (source code)

eclipse

MyApp (APK)

MyApp (.smali)

Smali patch

Cthulhu (APK)

Cthulhu (.smali)

Cthulhu (APK) +
evasion heuristic

Step b. – Inject an Evasion Heuristic

- Same procedure as in a. but the code we want to inject is a static VM evasion heuristic

- Simply check the **`Build.DEVICE`** field to find out if app is running on Emulator

```
String device = Build.DEVICE;
if (device.equals("generic")) {
       String env = "Emulator";
}
else {
        String env = "Device";
}
```

Step c. – Verify the repackaged app

- Submit both the original and the repackaged app on an online analysis service
  - (*e.g.*, Andrubis)

- Compare the produced reports

HINTS & TIPS

- Always ensure you have sufficient amount of registers when patching **.locals**
- Always fix the package name path in any injected method call

```
invoke-virtual p0, Lcom/example/myapp/StartActivity;->
getApplicationContext()Landroid/content/Context;
```

↓

```
invoke-virtual p0, Lcom/example/cthulhu/MainActivity;->
getApplicationContext()Landroid/content/Context;
```