JEB
互动的Android反编译

JEB是一个功能强大的为安全专业人士设计的Android应用程序的反编译。
反向工程或审计APK文件，并减少许多工程师的分析时间。

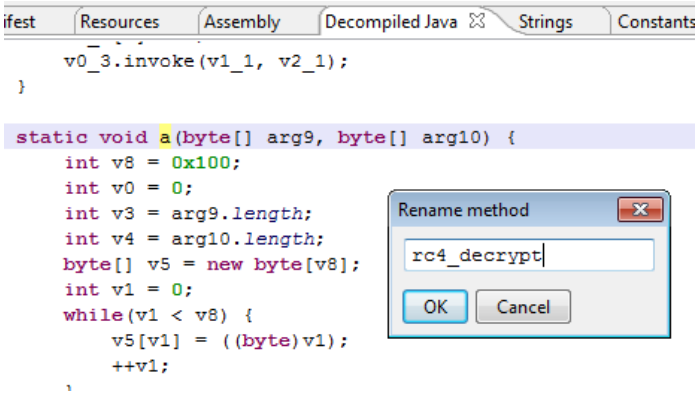
```
class TraceDisplayInformation {
    private CoordinatesE6 center;
    private int zoom;

    public TraceDisplayInformation(CoordinatesE6 arg
        super();
        int v1 = arg4.get_lng();
        int v2 = arg4.get_lat();
        this.center = new CoordinatesE6(v1, v2);
        this.zoom = arg5;
    }

    public CoordinatesE6 get_center() {
        return this.center;
```

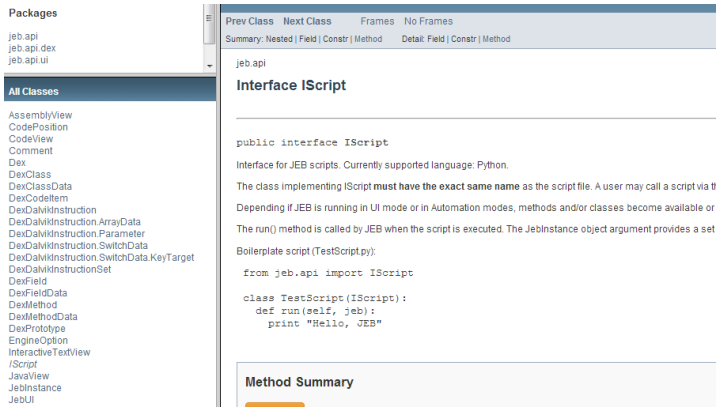
强大

JEB的独特功能是，其Dalvik字节码反编译为Java源代码的能力。无需DEX-JAR转换工具。我们公司内部的反编译器需要考虑的Dalvik的细微之处，并明智地使用目前在DEX文件的元数据。

灵活

分析师需要灵活的工具，特别是当他们处理混淆的或受保护的代码块。JEB的强大的用户界面，使您可以检查交叉引用，重命名的方法，属性，类，代码和数据之间导航，做笔记，添加注释，以及更多。





可扩展

利用API来使用脚本和插件扩展JEB， 例如：访问反编译的Java程序的AST来去除混淆层；利用非互动JEB来自动化后端处理。

API提供的语言：Python, Java.

JEB
互动的Android反编译

JEB比现有工具的两个主要优点是交互性和灵活性，以及工业级的反编译器输出。他们允许工程师分析，逐渐了解复杂的代码块。

```
public class Crypto
{
  public static void rc4_crypt(byte[] paramArrayOfByte1, byte[] paramArray
  {
    int i = paramArrayOfByte1.length;
    int j = paramArrayOfByte2.length;
    byte[] arrayOfByte = new byte[256];
    int k = 0;
    int m;
    int n;
    label30: int i2;
    int i3;
    if (k >= 256)
    {
      m = 0;
      n = 0;
      if (n < 256)
        break label68;
      i2 = 0;
      i3 = 0;
    }
    for (int i4 = 0; ; i4++)
    {
      if (i4 >= j)
      {
        return;
        arrayOfByte[k] = ((byte)k);
        k++;
        break;
```

第三方Java反编译器输出（左）

- 静态代码，没有互动性
- 反编译错误（箭头）
- 结果在不可读性和可用性差

```
public static void rc4_crypt(byte[] key, byte[] data) {
    int v10 = 0x100;
    int keylen = key.length;
    int datalen = data.length;
    byte[] sbox = new byte[v10];
    int i = 0;
    while(i < v10) {
        sbox[i] = ((byte)i);
        ++i;
    }

    int k = 0;
    i = 0;
    while(i < v10) {
        k = (sbox[i] + k + key[i % keylen]) % 0x100 & 0xFF;
        byte v7 = sbox[i];
        sbox[i] = sbox[k];
        sbox[k] = v7;
        ++i;
    }

    i = 0;
    k = 0;
    int j = 0;
    while(j < datalen) {
        i = (i + 1) % 0x100 & 0xFF;
        k = (sbox[i] + k) % 0x100 & 0xFF;
        v7 = sbox[i];
        sbox[i] = sbox[k];
        sbox[k] = v7;
        data[j] = ((byte)(data[j] ^ sbox[(sbox[i] + sbox[k]) % 0x100 & 0xFF]));
        ++j;
```

工程师进行了分析后的JEB的输出（右）代码。

该方法的代码是整齐的结构化和可读性。

更多的例子在我们的网站上。

请在我们的网页上询问试用版并且了解更多有关定价细节。