JEB
インタラクティブ Android デコンパイラ

JEBはセキュリティの専門家のために設計されたパワフルなAndroidアプリ用デコンパイラです。リバースエンジニアリングによるAPKファイルの監査や解析に要する時間を短縮することができます。

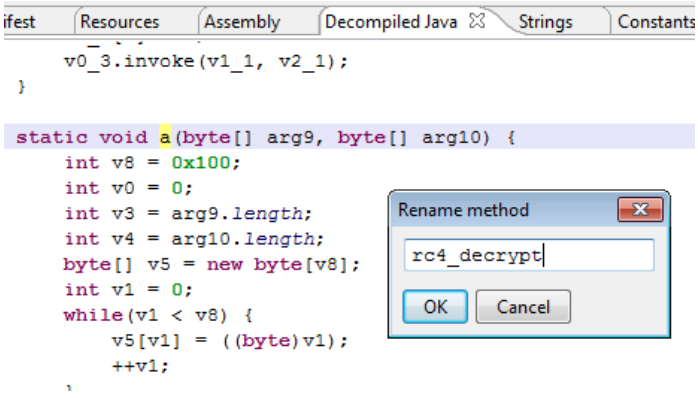
```
class TraceDisplayInformation {
    private CoordinatesE6 center;
    private int zoom;

    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;
```

本格的なDalvikデコンパイラ
JEBの特徴はDalvikのバイトコードをJavaのソースコードに直接デコンパイルできることです。dexからjarへの変換ツールは必要としません。また、当社のデコンパイラはDEXファイルのメタデータを利用することで、Dalvik固有の特徴を考慮しています。

インタラクティビティ
特に難読化されたコードの解析を行う際には、解析者にとって柔軟なツールはとても重要になります。JEBのパワフルなユーザインターフェイスは相互参照によるコード間の移動、メソッド・フィールド・クラス名のリネーム、コード-データ間のナビゲートといった様々な機能を提供します。



APKファイルの総合的な情報の閲覧
実行コードだけではなく、リソース、assets、証明書、文字列、定数値などの情報も調査することができます。

調査履歴の追跡
JEBは調査に要する時間を無駄にはしません。解析結果をJEBデータベース形式で保存し、JEBリビジョン管理機構により調査履歴の追跡が可能です。

マルチプラットフォーム
JEBはWindows、Linux、Mac OS上で動作します。

JEB
インタラクティブ Android デコンパイラ

既存のツールに対するJEBのアドバンテージは、産業グレードのデコンパイル結果を出力できるというだけではなく、インタラクティビティとフレキシビリティの2点にあります。それらは、解析者が複雑なコードを理解し、解析する際の大きな助けとなります。

```
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;
```

他社のデコンパイル結果（左図）

- インタラクティブに操作できない静的なコード
- 正しくデコンパイル出来ない箇所
- 可読性が低くユーザビリティが低い出力結果

```
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;
```

解析者による解析後のJEBの出力(右図)

- きちんと構造化されたコード出力
- リネーム機能による高い可読性の実現

更に具体例をご覧になりたい方は、私共のウェブサイトをご覧ください。

ライセンスや価格の詳細を含め、より詳細な情報は私共のウェブサイトをご覧ください。