JEB
대화형 안드로이드 디컴파일러

JEB은 보안 전문가를 위해 만들어진 강력한 안드로이드 앱 디컴파일러입니다. APK 파일을 역공학 또는 감사해야하는 엔지니어의 분석 시간은 반이상으로 줄일수 있습니다.

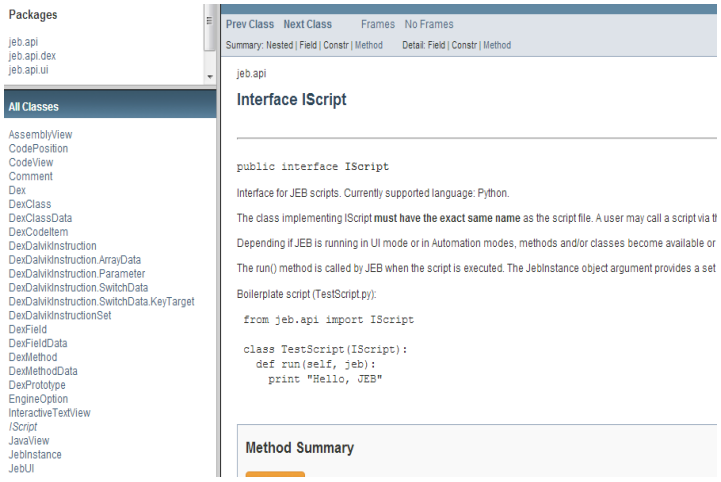
```
class TraceDisplayInformation {
    private CoordinatesE6 center;
    private int zoom;

    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;
```

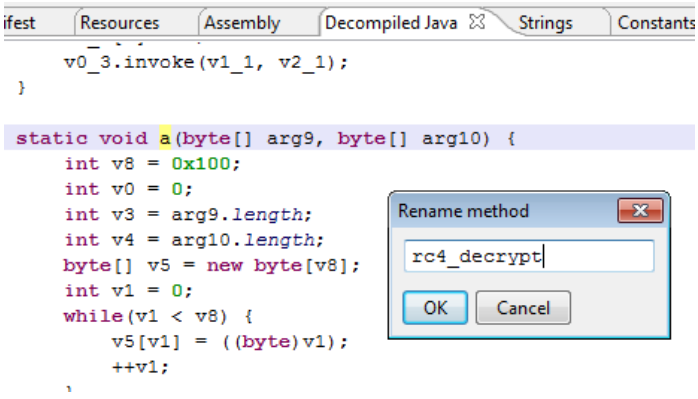
대화형식

분석가들이 난독화 또는 보호된 코드를 다루기위해서 유연한 도구를 필요로 합니다. JEB의 강력한 UI는 사용자가 크로스레퍼런스를 찾을수 있고, 메소드, 필드, 클래스 이름을 바꿀수있고, 코드및 데이타를 탐색할수 있으면, 주석을 달을수 있습니다.



완전한 달빅(Dalvik) 디컴파일러

JEB의 기능은 자바 소스코드를 달빅(Dalvik) 바이트코드로 디컴파일하는 기능입니다. dex-to-jar 변환 도구가 필요없습니다. 자체 개발된 디컴파일러는 달빅의 미묘한점까지 고려했으며, DEX 파일에 포함되 메타 데이타까지 유용하게 활용하고 있습니다.



전체 APK 뷰

압축 해제 된 리소스와 자산, 인증서, 문자열 상수 등을 검토가능함.

진행 상황을 추적 가능함

분석가의 시간을 낭비하지 마십시오. JEB 데이터베이스 파일에 분석을 저장하고 JEB의 수정 내역을 통해 진행 상황을 추적 할 수 있습니다.

다중 플랫폼

JEB은 윈도우, 리눅스 및 Mac OS에서 실행됩니다.

JEB
대화형 안드로이드 디컴파일러

기존 도구 들에 비해 JEB의 두 주요 장점은 대화형작업과 유연성을 비롯하여 회사용급 디컴파일러 결과물입니다. 이 장점들은 분석가들이 복잡한 코드를 분석하고 점차적으로 이해 할 수 있게 합니다.

```
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;
```

타사 자바 디컴파일러 결과 (왼쪽)

- 정적 코드, 비 대화형 작업
- 디컴파일러 오류 (화살표)
- 읽기 힘들고, 사용하기 불편함

JEB의 출력 (오른쪽), 엔지니어에 의해 분석된 후.

방법 코드가 깔끔하게 구조화되있고 읽기 쉽게되 있습니다.

저희 웹 사이트에 더 많은 예제가 있습니다.

```
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;
```

저희 웹 사이트에서 라이센스 및 가격 세부 사항을 포함한 자세한 내용을 알아보십시오.