Java Reverse Engineering

Easy enough... or is it?

How does Java work?

- .JAR files
  - This is the Java “executable”
  - Just fancy zip file renamed as .jar
  - Contains compiled class files and metadata

- .class files
  - Represents a .java file
  - Contains JVM bytecode

- metadata:
  - In file META-INF/MANIFEST.MF
  - Tells the JVM which class has the main method

How do we reverse engineer Java?

- Decompilers
  - Turn bytecode back into Java (VERY accurately)
  - Examples: Procyon, FernFlower, JD-GUI, etc
  - Break when program is obfuscated
    - Obfuscators rename variables, add code that is never run, encrypt strings, etc.
    - Popular obfuscators are ZKM, Allatori, Proguard

- Deobfuscator
  - Attempt to reverse obfuscator's destruction
  - May make it possible to use a decompiler after
  - Cannot recover all information: variable/method names lost
  - Examples: https://github.com/java-deobfuscator/deobfuscator, (https://mega.nz/file/H0wCXIrK#7Ud5O9-4RmZi5riLizvLByv-ET67PU4KfVlMXo-jG4o) https://github.com/java-deobfuscator/deobfuscator-gui

Recommendations

- By default, use Bytecodeviewer
  - Easy to use
  - Includes several decompilers and bytecode viewers
  - Can decompile .apk
  - https://bytecodeviewer.com

- If obfuscated, use deobfuscator-gui
  - Can deobfuscate programs obfuscated by commercial obfuscators (most of them)
  - https://github.com/java-deobfuscator/deobfuscator-gui
  - If this doesn't work, or if the jar is custom obfuscated:
    - Use ObjectWeb ASM library to manipulate the bytecode yourself. (This sucks to do)

EXAMPLE TIME!