Code obfuscation techniques

Eric Lafortune

CTO at GuardSquare
Creator of ProGuard and DexGuard
Topics

- Threats, attacks, and techniques
- Android applications and build process
- Obfuscation techniques
Threats

Key loggers
Spyware
Malware
Advertisements
Secure communication
Authentication
Keys
IP
App
Attackers
Time
Choice
Static analysis

- Disassemblers: dexdump, baksmali
- Decompilers: dex2jar + jad, JD-GUI, JEB,…
- Resources: aapt, apktool,…
Dynamic analysis

- Debuggers: adb,…
- Subverted runtime
- Cracking tools
Android application

**.apk**

- Manifest
  - (AndroidManifest.xml)
- Asset files
  - (assets/*)
- Resource constants
  - (resources.arsc)
- Resource files
  - (res/*/*)
- Dalvik bytecode
  - (classes.dex)
- Native libraries
  - (lib/*/*.so)
Android build process

- Application Java source
  - Java source
  - Libraries
    - Java bytecode
  - Native source
  - Compiled XML resource
  - Compiled bytecode
  - Dalvik bytecode
  - Native libraries
  - Signatures

- Application bytecode
  - ProGuard
  - Dex
  - Jar signer
  - Zip align

- Assets
  - Compiled bytecode
  - Dalvik bytecode
  - Native libraries
  - Signatures
ProGuard

Application code

Libraries

Android runtime

Shrink

Optimize

Obfuscate

Processed code

Android runtime
Shrinking

- Classes, fields, methods
Optimization

At the bytecode instruction level:

- Dead code elimination
- Constant propagation
- Method inlining
- Class merging
- Remove logging code
- Peephole optimizations
- Devirtualization
- ...

Obfuscation

Traditional name obfuscation:

- Rename identifiers:
  class/field/method names

- Remove debug information:
  line numbers, local variable names,...
Name obfuscation

```java
public class MyComputationClass {
    private MySettings settings;
    private MyAlgorithm algorithm;
    private int answer;

    public int computeAnswer(int input) {
        ...
        return answer;
    }
}
```
Name obfuscation

```java
public class MyComputationClass {
    private MySettings settings;
    private MyAlgorithm algorithm;
    private int answer;

    public int computeAnswer(int input) {
        ...
        return answer;
    }
}
```

```java
public class a {
    private b a;
    private c b;
    private int c;

    public int a(int a) {
        ...
        return c;
    }
}
```
public class MyVerificationClass {
    public int checkSignatures() {
        ...
        return activity
            .getPackageManager()
            .checkSignatures("mypackage1", "mypackage2");
    }
}
public class MyVerificationClass {
    public int checkSignatures() {
        ...
        return activity
            .getPackageManager()
            .checkSignatures("mypackage1", "mypackage2");
    }
}

public class a {
    public int a() {
        ...
        return a
            .getPackageManager()
            .checkSignatures("mypackage1", "mypackage2");
    }
}
Server-side code

Client-side application

Server-side application
String encryption

String KEY = "Secret key";

String KEY = new String(Base64.decode("U2VjcmV0IGtleQo="));
Reflection

System.out.println("Hello world!");

Class clazz =
    Class.forName("java.io.PrintStream");

Method method =
    clazz.getMethod("println",
        new Class[]{String.class});

method.invoke(nul, new Object[]{"Hello world!"]);
Control flow obfuscation
Control flow flattening
Opaque predicates

boolean flag = true;

boolean flag = System.currentTimeMillis() > 0;
Arithmetic obfuscation

```c
int c = a + b;
int c = (a ^ b) + 2 * (a & b);
```
Dynamic class loading

Native code

Java Native Interface

classes.dex

classes.dex

libutil.so
Data encryption

- **Standard cryptography APIs**

```java
Cipher cipher = Cipher.getInstance("AES/CFB/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE,
            key, initializationVector);
byte[] decrypted = cipher.doFinal(encrypted);
```

- **SQLCipher**
White-box cryptography

Research: www.whiteboxcrypto.com

```java
final int KEY = 42;

int decryptValue(int encryptedValue) {
    return encryptedValue * KEY;
}
```

Real examples: DES, AES
Application checks

- **Check application certificate:**

```java
byte[] certificateData = activity
    .getPackageManager()
    .getPackageInfo(activity.getPackageName(), PackageManager.GET_SIGNATURES)
    .signatures[0]
    .toByteArray();
```

- **Check debug flag:**

```java
boolean debug =
    (activity.getApplicationInfo().flags & ApplicationInfo.FLAG_DEBUGGABLE) != 0;
```
Environment checks

- Emulator detection
- Root detection
Packing

Files

Dalvik bytecode
(classes.dex)

Memory

Dalvik bytecode
Summary

Nothing is unbreakable, but you can raise the bar:

- String encryption
- Reflection
- Control flow obfuscation
- Dynamic class loading
- Native code
- Data encryption
- Application checks
- Environment checks
- Packing